VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Wei Lu

DESIGN AND IMPLEMENTATION OF AUTONOMOUS STAIR CLIMBING WITH NAO HUMANOID ROBOT

Technology and Communication

2015

FOREWORD

So far, it has been three years since I started my Bachelor's degree at Vaasan Ammattikorkeakoulu, University of Applied Sciences. I would like to express my greatest gratitude to VAMK for giving me such a peaceful environment to study.

Moreover, my thesis would not have been possible without the support and encouragement by many people. First of all, I would like to give my greatest appreciation to my supervisor Dr. Yang Liu for giving me the opportunity to study in an exciting field of research. With his experience, he guided my final thesis scientifically and provided plenty of helpful suggestions along the way.

In addition, I would like to give my great appreciation to Dr. Smail Menani. Without the help of him, my final thesis cannot complete at the high level.

Additionally, I would like to thank to all the staff in Vaasan Ammattikorkeakoulu, University of Applied Sciences including Dr. Ghodrat Moghadampour, Dr. Chao Gao, Mr. Jukka Matila, Mr. Santiago Chavez Vega, Mr. Jani Ahvonen and the other teachers who taught me and gave me so many helps.

Finally, I would like to thank all previous and current fellows in the Robotics Lab for the great atmosphere and work environment and great encouragement including Lin Han. I sincerely hope that my fellows can gain more accomplishment in the future.

Lu Wei

Vaasa, Finland

2/5/2015

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Wei Lu
Title	Design and Implementation of Autonomous Stair Climbing with Nao Humanoid Robot
Year	2015
Language	English
Pages	75 + 1 Appendices
Name of Supervisor	Yang Liu

With the development of humanoid robots, autonomous stair climbing is an important capability. Humanoid robots will play an important role in helping people tackle some basic problems in the future.

The main contribution of this thesis is that the NAO humanoid robot can climb the spiral staircase autonomously. In the vision module, the algorithm of image filtering and detecting the contours of the stair contributes to calculating the location of the stairs accurately. Additionally, the strategy of climbing the spiral staircase is based on the vision system. Moreover, the mathematical model is utilized to calculate the angle of the stairs and the distance between the NAO humanoid robot and the each stair. In addition, animations of climbing the stairs are established by using key frames.

In this thesis, 5th generation NAO robot was used for developing this application. This application was mainly developed by using Python programming language based on Windows 8 operating system and Naoqi 2.1 operating system. Moreover, this application was simulated by Webots at Windows 8 operating system. In addition, Open Source Computer Vision Library (OpenCV) was used for processing image. Choregraphe 2.1.2 was used for designing the behavior of climbing the stairs. In this thesis, the implementation methods were definition, analysis, design, implementation and testing.

In conclusion, the humanoid NAO robot has powerful functionality and humanoid robot will become more and more popular in the future. In this project, the robot achieves the capability of climbing the spiral staircase autonomously.

Key words Humanoid NAO robot, Python, OpenCV, behavior design

CONTENTS

LIST OF ABBREVIATIONS

RoboCup	Robot Soccer World Cup
SSID	Service Set Identifier
FTP	File Transfer Protocol
OpenCV	Open Source Computer Vision Library
SDK	Software Development Kit
PC	Personal Computer
CPU	Central Processing Unit
IDE	Integrated development environment
PIL	Python Imaging Library
RGB	Red, green and blue
HSV	Hue, Saturation and Value
VGA	Video Graphics Array
FPS	frames per second

LIST OF FIGURES AND TABLES

LIST OF TABLES

LIST OF APPENDICES

1 INTRODUCTION

1.1 Purpose of the Thesis

The thesis introduces a python application with NAO humanoid robot. The main purpose of this thesis is to make NAO humanoid robot climb the stairs autonomously.

1.2 Overview Structure of the Thesis

The thesis consists of nine chapters. The first chapter describes the information of the background, such as stair definition, NAO robot and motivation. The second chapter introduces the overall structure of the thesis. The third chapter introduces the relevant technology used in this thesis including used software and the programming language. The fourth chapter introduces the vision system of NAO robot including hardware of NAO robot, color space, image acquisition, cutting image, detecting white and red paper, image filtering and finding contours. The fifth chapter focuses on the behavior design of climbing the robot. The sixth chapter introduces the strategy. The NAO robot can calculate steps and rotate its body directly to the stair at the proper position by using this strategy. The seventh chapter focuses on the troubleshooting including two ways of detecting the stairs and improving the movement method. The eighth chapter introduces the future research. The ninth chapter is a conclusion of the whole thesis.

1.3 Stair Definition

In this project, there are three stairs in one staircase. Here are some attributes of staircase.

- Height of one stair is 4.7cm.
- Width of one stair is 17cm.
- Each stair has one white rectangle paper which is in the front of the stair

- The third stair has also one red rectangle paper which is at the end of the stair.

- The angle between first step and second step is 15^{o}

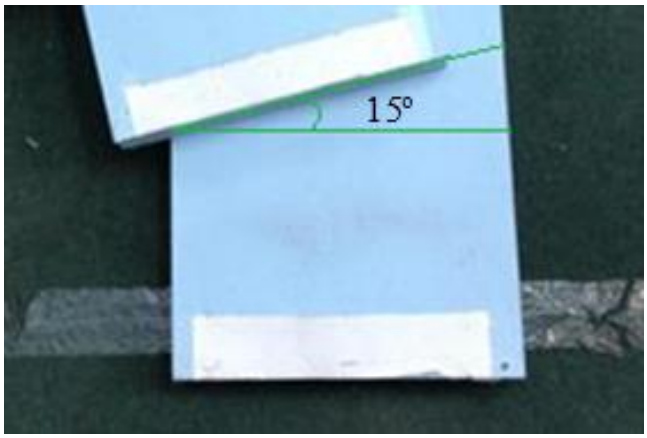

Figure 1. The degree between first step and second step

- The angle between second step and third step is 25^{o}

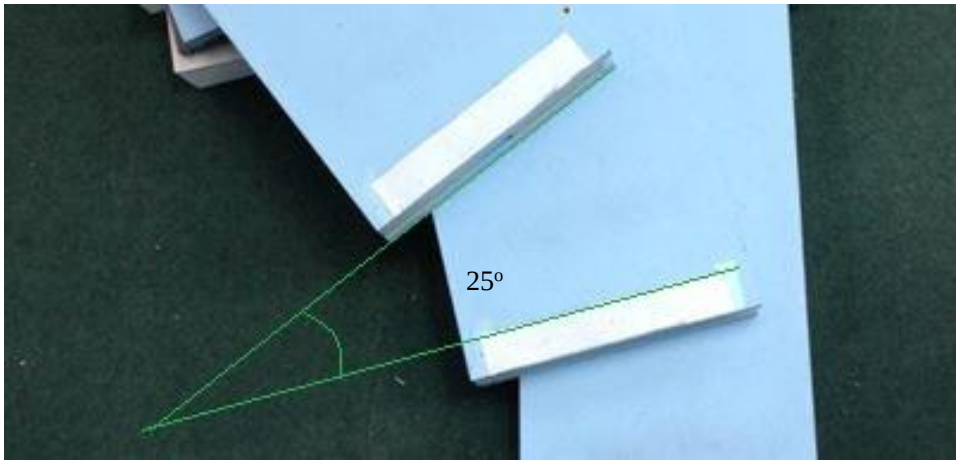

Figure 2. The angle between second step and third step

Here is the picture of staircase

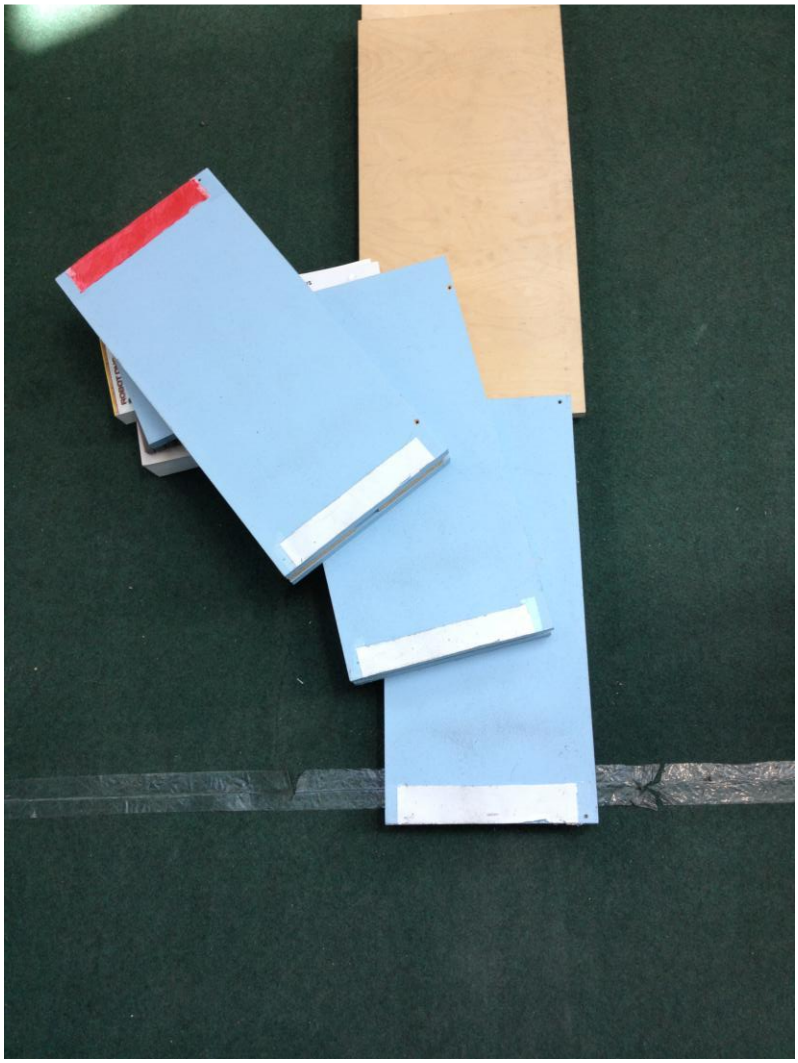

Figure 3. Overview of staircase

1.4 Introduction to NAO Robot

1.4.1 History

The NAO humanoid robot which is developed by Aldebaran Robotics is programmable, open architecture robot. NAO is being used all over the world for educational and research purposes, from primary school through to university in over 480 universities. /1/

Figure 4. Humanoid NAO robot /2/

The project of the NAO began in 2005. The NAO robot replaced Sony's Aibo (puppy robot) and was chosen to be the official platform of RoboCup (Robot Soccer World Cup) in August 2007. It has been used since 2008 at the competition in Suzhou, China. In 2009, 24 teams from all over the world participated in the RoboCup competition using a total of 100 NAOs./3/

Figure 5. NAO robot in RoboCup (Robot Soccer World Cup)

1.4.2 Construction of NAO robot

Table 1. Construction of NAO robot

NAO (V5) Construction/4//5/	
Height (mm)	574
Depth (mm)	311
Width (mm)	275
Weight (kg)	5.4
Autonomy	1.5 hours for active use
Degrees of freedom	25
CPU	Intel Atom @ 1.6 GHz
Material	ABS-PC/PA-66/XCF-30
Built-in OS	NAOqi 2.0 (Linux-based)
Compatible OS	Windows, Mac OS, Linux
Programming languages	Python, C, C++, .Net, Java, MATLAB, Urbi.
Battery	Build in rechargeable battery
Connectivity	Ethernet, Wi-Fi

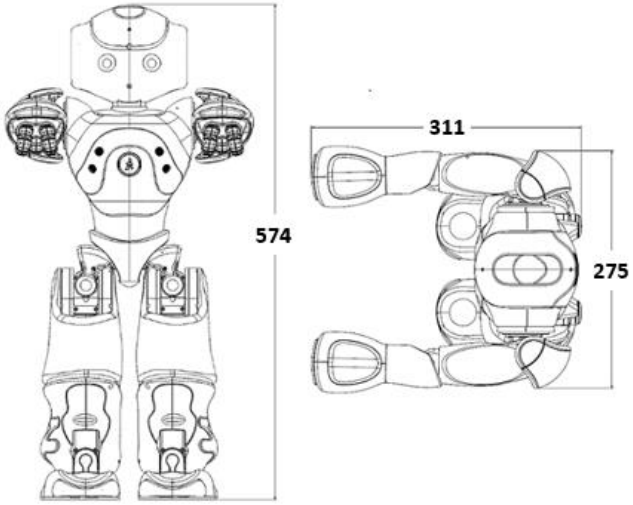

Figure 6. Dimension of NAO robot

1.5 Motivation

With the development of technology, there is no doubt that humanoid robots will become more and more popular and come to our daily life soon. The humanoid robots will play an important role in helping human beings to tackle some basic or even dangerous tasks.

For the humanoid robot, autonomous stair climbing is an important and basic capability. Meanwhile, the NAO robot has excellent functionality and performance. In addition, it will be a great platform to be used in the whole world.

Currently, the robot team in Botnia has made plenty of NAO robot applications such as picking up ball, playing the computer game called Candy Crush Saga and drawing the contours of objects. The field of humanoid robot is becoming more and more popular.

2 OVERALL STRUCTURE

This chapter introduces the overall structure of climbing the stairs with NAO humanoid robot and also the brief illustration of each component.

2.1 Overall Structure

The whole system contains several independent but interrelated modules shown in Figure 7. The whole system has five parts: vision system, behavior module, communication module, controller PC and strategy.

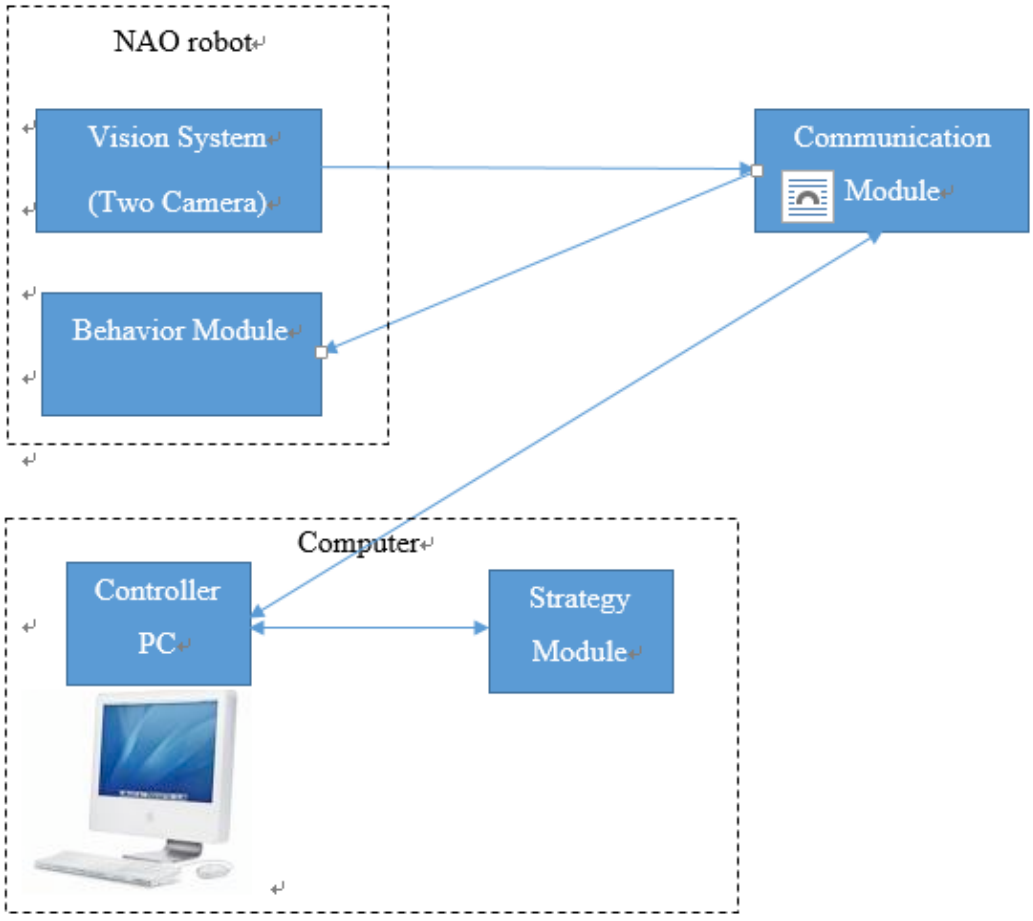

Figure 7. Overall structure of whole system

Here are the introductions of each part.

- Vision system

  The vision system consists of two cameras which are used for getting images and then sending these images to the controller PC through communication module. The introduction of vision system will be shown in Chapter 4 in details.

- Communication module

  Communication module is used for communication between controller PC and the NAO humanoid robot.

  As for the communication module, the NAO robot can communicate with a computer by a wired connection using an Ethernet cable or wireless connection using Wi-Fi. The wireless connection depends on the wired connection which can configure the parameter of wireless connection in order to make the NAO robot remember the SSID (Service Set Identifier) and the password of Wi-Fi.

  The NAO robot also supports FTP (File Transfer Protocol) service which can send files and receive files between the NAO robot and the remote PC by using the IP address of the NAO robot.

  After the computer connects the NAO robot by using an IP address and port, the value of which is “9559”, the Naoqi in Naoqi OS will start to run and load all the libraries automatically by using preferences file which is called “autoload.ini”. Then, all the modules in these libraries can be called by the broker. In this way, the NAO robot can run the program or do some animation which is designed by the developer.

  Figure 7 shows how the “Naoqi” works. /9/

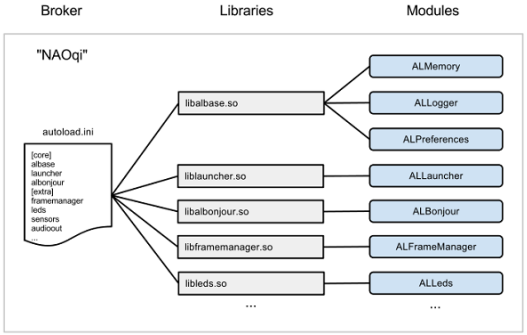

Figure 8. Communication part (1)

As is shown in Figure 8, the broker contains all the libraries and each library contains the modules. The broker acts as a controller which can access all the module in this tree or across network by finding their libraries.

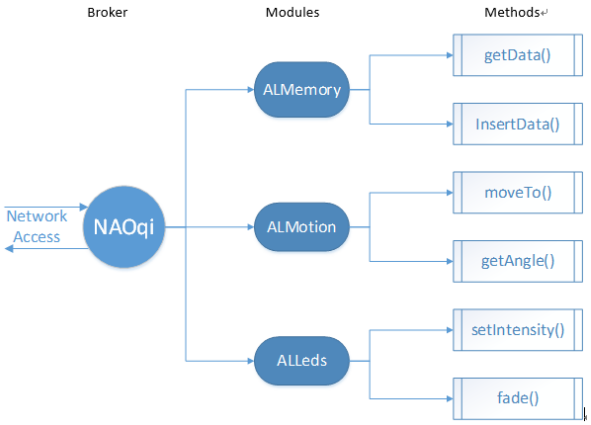

Figure 9. Communication part (2)

As shown in Figure 8, there are three parts: the broker, modules and methods. Broker is an object which can access to the network and access to the modules. Proxy is an object which behaves as the module which can call the methods in this module. Methods is the function with which the user can make the robot do according to the code such as making speech, movement and so on.

- PC controller

  PC controller is a computer which can call all the strategies and methods.
- Strategy

The strategy includes image processing and calculation of some angles and distance which is used for adjusting the NAO humanoid robot. The introduction of strategy will be shown in Chapter 6 in details.

- Behavior module

  Behavior module is the animation of climbing the stairs. The introduction of Behavior module will be shown in Chapter 5 in details.

2.2 Flow Char of the whole Project

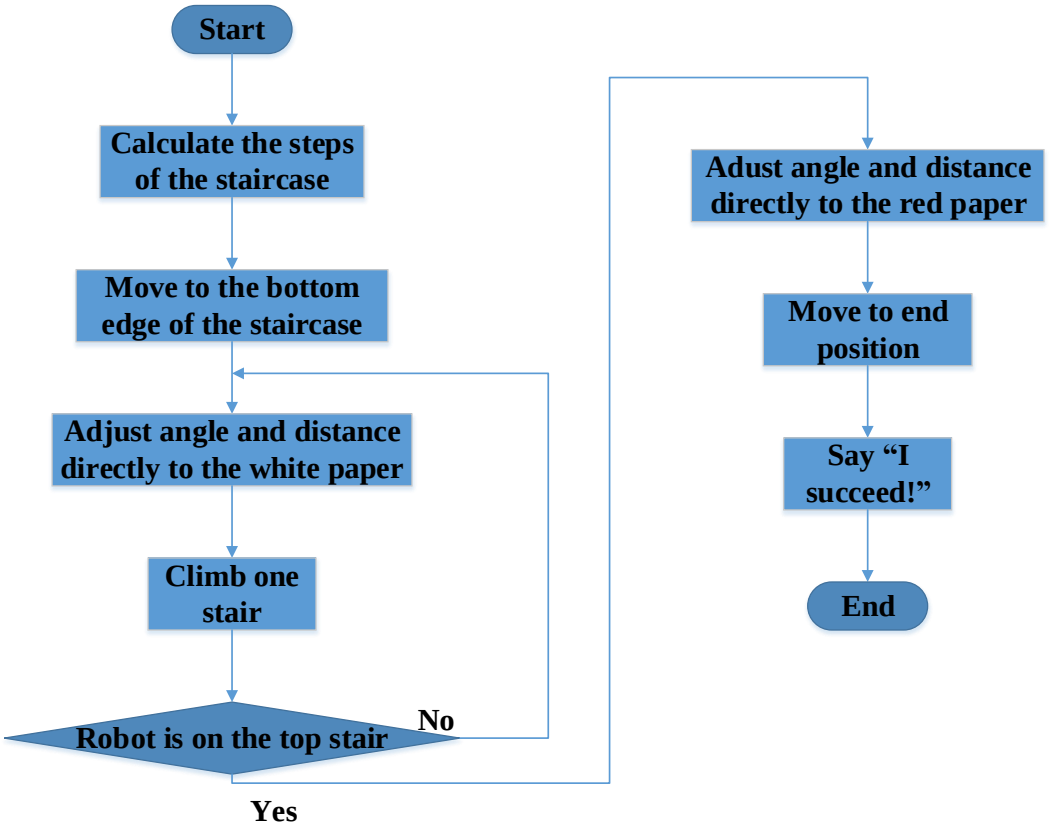

Figure 10. Flow chart of Whole Project

3 RELEVANT TECHNOLOGY

3.1 Used Software

In this part, it introduces the programming environment of the NAO robot. There are many softwares used in this thesis such as Choregrapghe 2.1.2, Webots 8.0.3, Naoqi, Python SDK(Software Development Kit) and Eclipse.

3.1.1 Choregraphe 2.1.2

Choregraphe is a multi-platform desktop application which can be installed in Windows, Linux and Mac operating system. Due to its friendly user interface, it is good for beginners to develop a NAO robot application. By using Choregraphe, it is easy to build NAO robot applications without writing the code.

Here are some features of Choregraphe:

- Drag-and-drop box
- Creating the dialogs and designing the animations and the behaviors
- Testing them on a real robot or a simulated robot which is inside the software.
- Monitoring and controlling the robot,
- Running the application by using the Python code.
- Directly presenting the sequence of the boxes in the program.

Here is the picture of Choregraphe 2.1.2

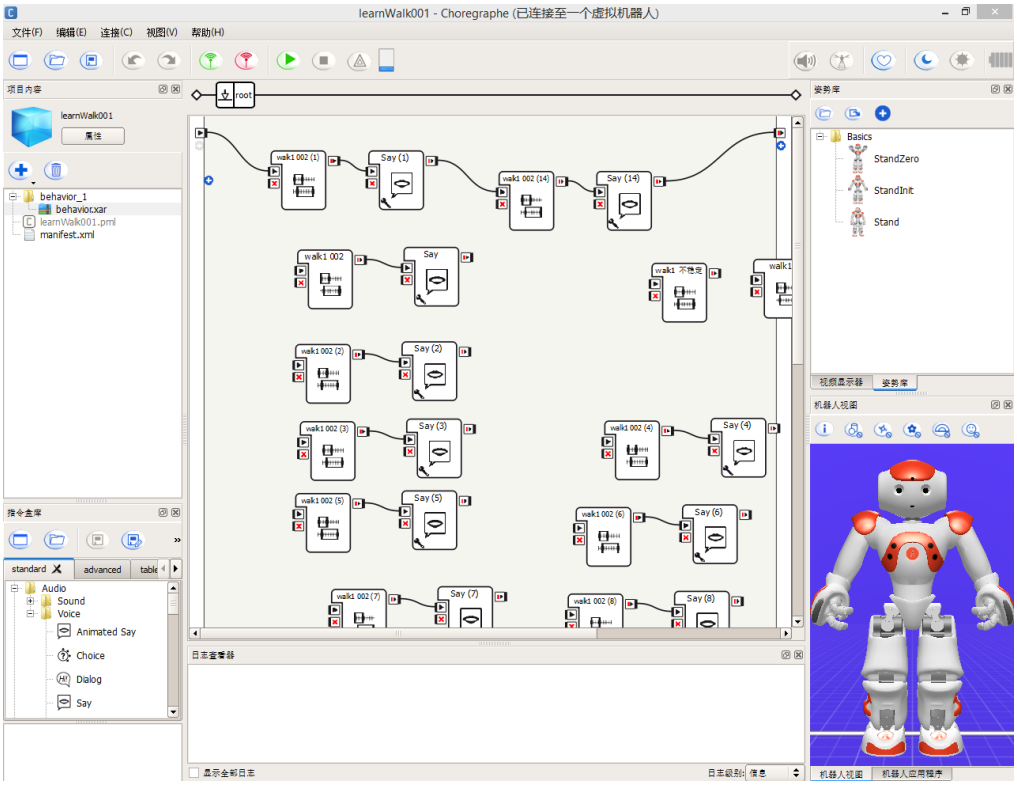

Figure 11. Choregraphe 2.1.2

In this thesis, Choregraphe 2.1.2 was used to design the animation for climbing the stairs.

3.1.2 Webots

Webots is a simulation software of NAO robot. When designing and testing the animation of NAO robot, developer always tests in webots first and then tests by using real NAO robot if everything is OK in order to protect the motor of the real NAO robot.

Here is the picture of Webots 8.0.3

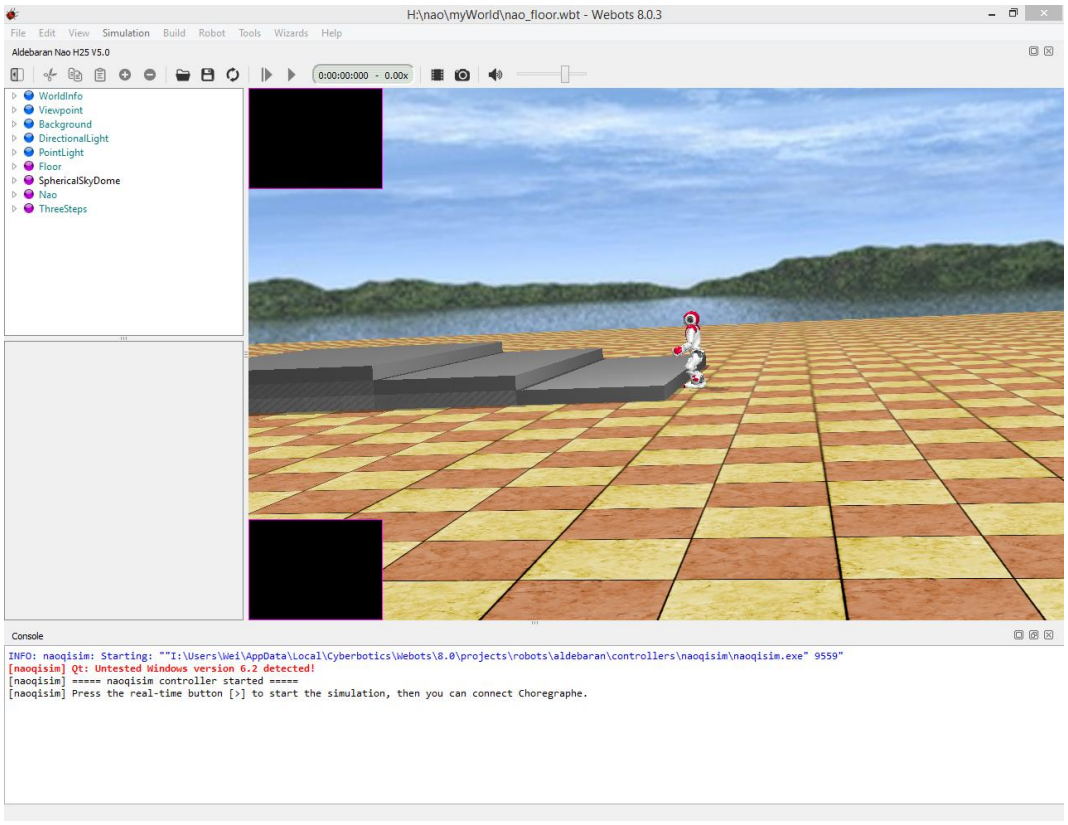

Figure 12 Webots 8.0.3

3.1.3 Naoqi OS and Naoqi

Naoqi OS is the operating system of the NAO robot which is an embedded GNU/Linux distribution based on Gentoo in order to fit the Aldebaran robot needs. Naoqi is the main software running on the robot under the Naoqi operating system distribution. Without Naoqi, no behaviors and actions can run.

3.1.4 Python SDK (Software Development Kit)

Python SDK is a programming package that enables the programmer to develop applications for the NAO robot. It can be used to create Python modules that run remotely or on the NAO robot.

Moreover, the CPU and memory of the NAO robot are limited, by using Python SDK, it will be possible to save the data to the remote PC. The remote PC will help the NAO robot handle the image processing and some large calculations.

3.1.5 Eclipse and PyDev

Eclipse is a powerful integrated development environment (IDE) which can be used to develop applications in many different programming languages such as Java, python, C, C++ and so on.

PyDev is a Python IDE or plugin for Eclipse, which can be used as Python IDE (Integrated development environment)

With the combination of eclipse and PyDev, it makes the program more efficient and effective. For example, it can import other modules automatically and show all the functions of the module just by using dot (.). Moreover, Eclipse has the capability of spell checking and grammar checking when the developer programs with Python.

3.2 Programming Language

Python is the programming language in the thesis project. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Compared with other programming languages such as C, C++ and Java, Python has so many advantages./6/

- Especially clean, straightforward syntax.
- Iterators, generators and comprehensions.
- Do not need to worry about that object's specific type.
- Huge standard library
- Great support for building web apps.

3.3 Used Module of Python

Table 2. The module of Python

Module	Description
ALProxy	An object that gives you access to all the methods or the module you are going to connect to.
opencv	A library of programming functions used for vision
PIL(Python Imaging Library)	A library which provides powerful image processing and graphics capabilities.
numpy	A package for scientific computing
matplotlib	A python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
time	A module which provides various time-related functions
almath	ALMath is an optimized mathematic toolbox for robotics.

3.4 Environment Configuration

In order to program the NAO robot with remote PC, three softwares are needed to configure which are opencv configuration and configuration of Eclipse and PyDev.

3.4.1 OpenCV Configuration

OpenCV (Open Source Computer Vision Library) is used to process image. OpenCV 2.4.11 was used in this thesis. The installation package can be found from the official website, http://opencv.org/downloads.html.

After running the installation package, it will extract all the files to a directory.

Then, the path of OpenCV "H:\Program Files (x86)\opencv\build\x86\vc10\bin" needs to add to system path.

Here are the steps for path configuration

- Open the control panel, then click System and Security and choose System
- Then, choose "Advanced system settings", it will show the system properties. Next, click "Environment Variables" button.
- The new user library which is called PATH and the content of PATH is "%PATH%;H:\Program Files (x86)\opencv\build\x86\vc10\bin"

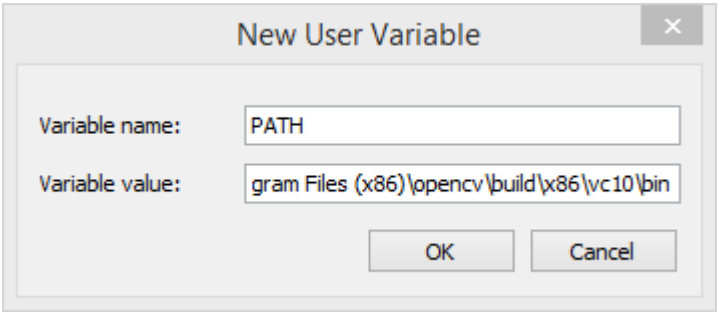

Figure 13. Configure the path for opencv

After the configuration of OpenCV, it also needs numpy-1.9.2 installation package and scipy-0.15.1 installation package.

The installation package numpy-1.9.2 can be found from the official website, http://sourceforge.net/projects/numpy/files/NumPy/.

The installation package scipy-0.15.1 can be found from the official website, http://sourceforge.net/projects/scipy/files/scipy/.

After installing these two library, OpenCV can be used in python.

The verification of OpenCV in python can use code “import cv2”, cv2 is one of the modules inside the OpenCV. If OpenCV is installed successfully, it will show nothing, otherwise, it will show the error message, which is “ImportError: No module named cv2.”

3.4.2 Configuration of Eclipse and PyDev

PyDev is a plugin of Eclipse. Therefore, Eclipse must be configured firstly and then install the PyDev.

As for Eclipse, the software Eclipse needs the support by JDK (Java Development Kit). Thus, if the eclipse cannot work, configuration of JDK (Java Development Kit) must be done before using Eclipse.

The configuration of JDK (Java Development Kit) is to install the JDK (Java Development Kit) and add the path of JDK (Java Development Kit) to the system variable PATH.

The verification of JDK can be tested by using command “cmd” and typing “java”, “javac” or “java –version”

After the configuration of Eclipse, click Help and then click “Install new Software”

Next, click “Add” button and type http://pydev.org/updates to the location and click OK and choose the PyDev to install as is show in Figure.

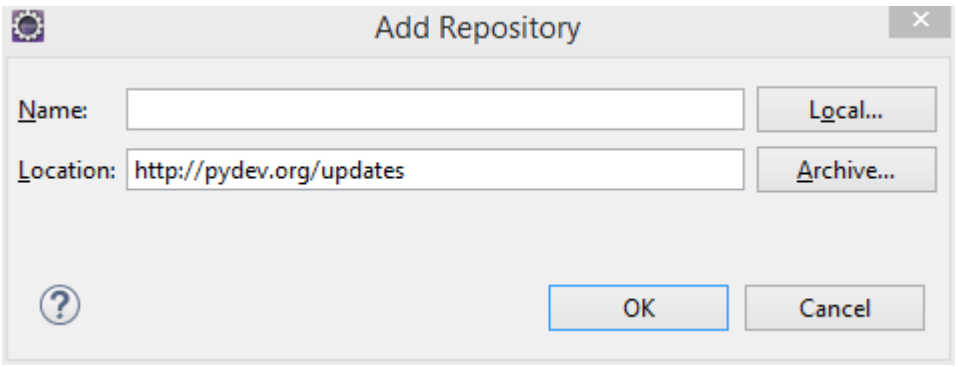

Figure 14. Configuration for PyDev (1)

After installing the PyDev. Go to Perferences—Pydev—Interpreters—Python Interpreter

Then click Quick-Auto-Config which will add the path of Python into Eclipse

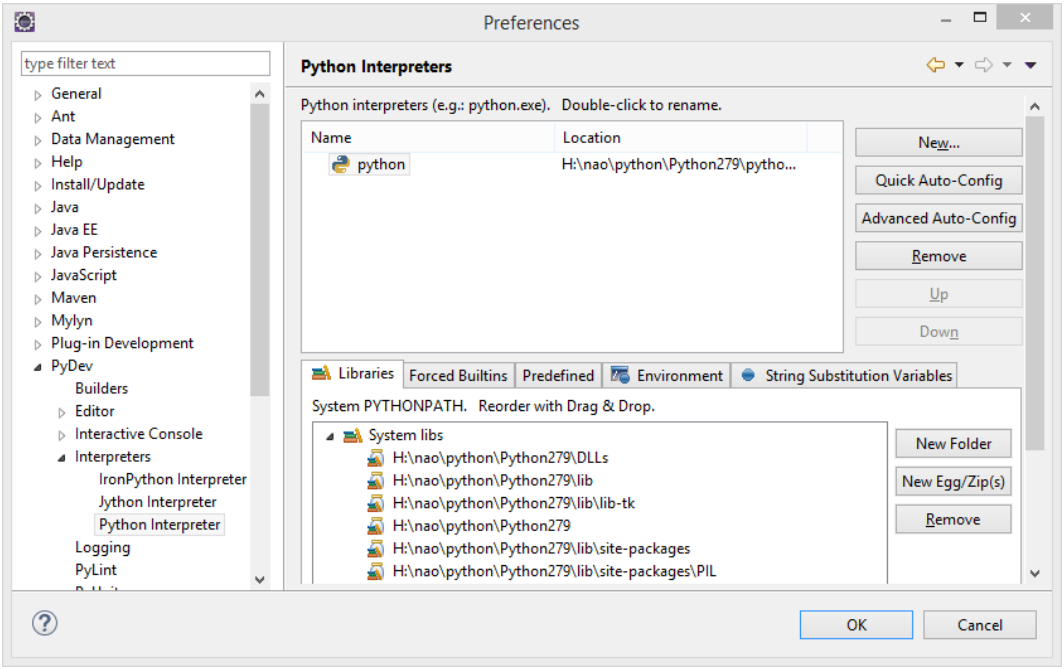

Figure 15. Configuration for PyDev (2)

As shown in Figure 27, Quick-Auto-Config button will add the path of Python and all the libraries into Eclipse. In this way, Eclipse and PyDev can be used to program with Python code

After the configuration of Eclipse and PyDev, it can create new PyDev project and PyDev package which can write Python code.

By using Eclipse and PyDev, the developers can see the Python code and the result at the same time.

Therefore, in this way, it will be more effective and efficient than the normal Python ide.

4 VISION SYSTEM

In order to make the NAO robot climb the stairs autonomously, the vision system of the NAO robot plays an important role in the detection of staircase and adjusting itself at the proper position so that the NAO robot can climb the stairs successfully.

In the project, as for the vision system, vision detection is based on color detection because color detection is more precise than shape detection and shape detection will have more noises. The details of the differences between detecting the stair by shape and detecting the stair by color will be shown in Chapter 6. Next, the NAO robot needs to get an image from its bottom camera, then cut the image to a small size in order to avoid the influence from its body because its hands and feet have the same color as the white paper which is on the stairs. Then, after image processing, the NAO robot detects white rectangle paper in order to calculate the rotation angle of the NAO robot and distance between the NAO robot and stairs. When the NAO is at the top stairs, the NAO robot detects a red rectangle paper to the end position.

4.1 Flow char of vision system

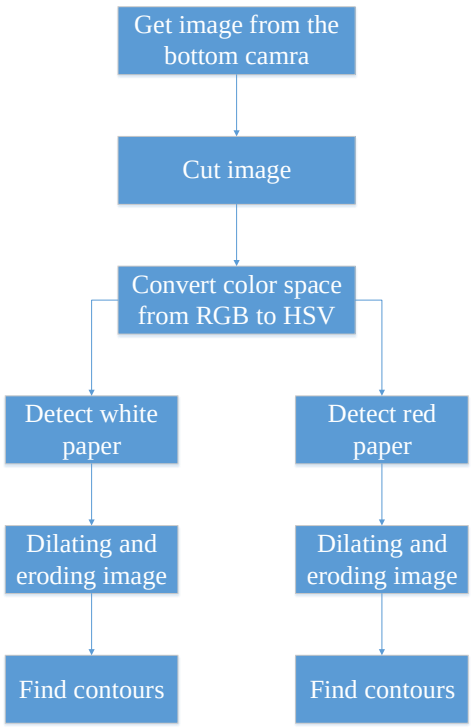

Figure 16. Flow chart of vision system

4.2 Hardware of NAO Robot

The NAO robot has two high performance cameras (top camera and bottom camera) for its vision system which are located in its forehead. Both of them have up to 1280*960 resolution and take 30 pictures for one second. There are six different resolutions which are 1280*960px, 640*480px, 320*240px, 160*120px, 80*60px, 40*30px.

In the project, the resolution is 640*480px.

Here is the range of two camera of the NAO robot

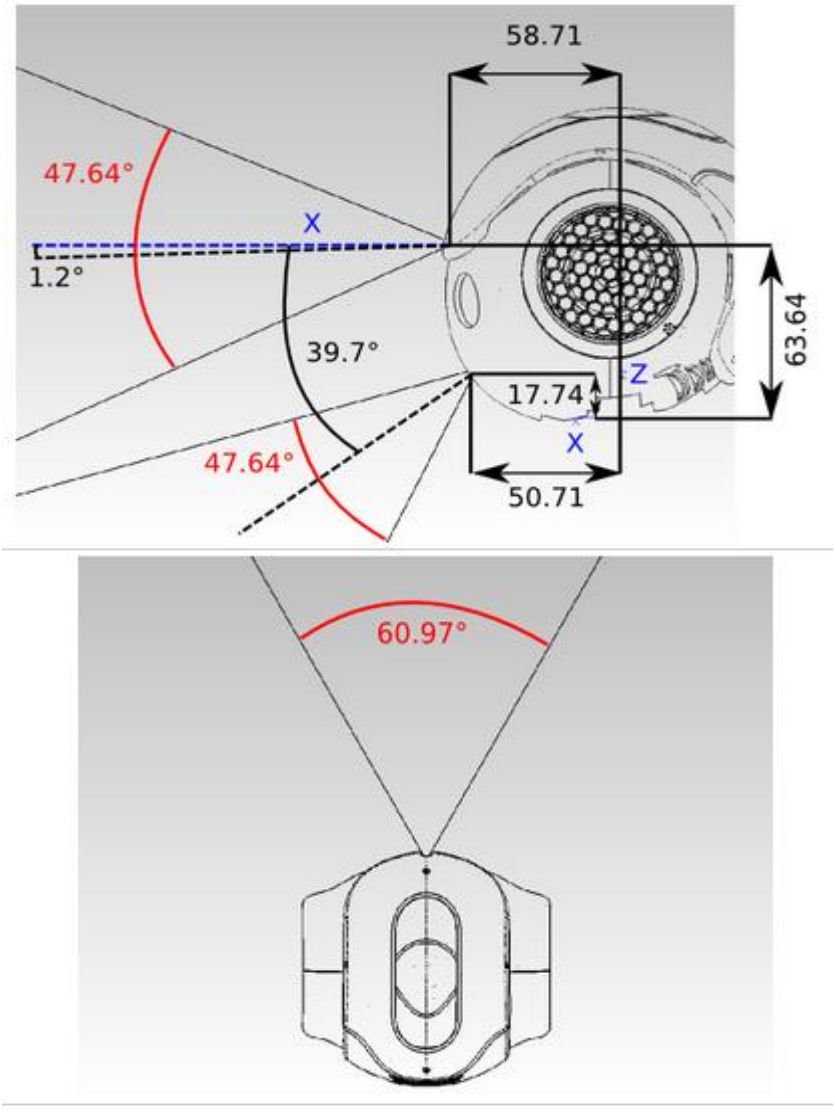

Figure 17. The range of two camera /7/

Here is the specification of the two cameras.

Table 3. The specification of cameras.

Camera	Model	MT9M114
	Type	SOC Image Sensor
Imaging Array	Resolution	1.22 Mp
	Optical format	1/6 inch
	Active Pixels (HxV)	1288x968
Sensitivity	Pixel size	1.9µm*1.9µm
	Dynamic range	70 dB
	Signal/Noise ratio (max)	37dB
	Responsivity	2.24V/Lux-sec (550 nm)
Output	Camera output	1280*960@30fps
	Data Format	(YUV422 color space)
	Shutter type	Electronic Rolling shutter (ERS)
View	Field of view	72.6°DFOV (60.9°HFOV,47.6°VFOV)
	Focus range	30cm ~ infinity
	Focus type	Fixed focus

4.3 Color Space

A color space is a model which is used for representing color as an ordered list of integer numbers. RGB and HSV were used in this project. RGB color space was used for saving images from the camera of the NAO robot and HSV color space was used to find a certain color paper in order to find each stairs and the end position.

Here is a list of supported color spaces of the NAO robot

Table 4. Supported color spaces

Name of color space	Description
YUV	Describe color by using luminance component and two chrominance components
RGB	Describe color by using red, green and blue
HSV	Describe color by using hue, saturation and value
HSY	Describe color by using hue, saturation and lightness

In this project, RGB (red, green and blue) color space was used to save image from the camera of the NAO robot. HSV (hue, saturation and value) color space was used to detect the color the white rectangle paper and red rectangle paper.

4.3.1 RGB Color Model

The RGB color model is an additive color model which is used for specifying colors. The RGB color format can represent any standard color or brightness using a combination of Red, Green and Blue components. This model shows the density of three color (red, green blue) by using number from 0 to 255. The number "0" represents minimum intensity and the number "255" represents maximum intensity

In Python, RGB value = [red, green, blue]

Here are some example colors with the intensity values of red, green and blue,

Table 5. Example color by using RGB color space /11/

Color	HTML / CSS Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
	Red	#FF0000	(255,0,0)
	Lime	#00FF00	(0,255,0)
	Blue	#0000FF	(0,0,255)
	Yellow	#FFFF00	(255,255,0)
	Cyan / Aqua	#00FFFF	(0,255,255)
	Magenta / Fuchsia	#FF00FF	(255,0,255)
	Silver	#C0C0C0	(192,192,192)
	Gray	#808080	(128,128,128)
	Maroon	#800000	(128,0,0)
	Olive	#808000	(128,128,0)
	Green	#008000	(0,128,0)
	Purple	#800080	(128,0,128)

4.3.2 HSV Color Space

HSV (Hue, Saturation and Value) is a type of color space. It has three elements: hue, saturation and value. /12/

- Hue is described as a range of number from 0 to 360 degrees which can represent hues of red (0), yellow (60), green (120), cyan (180), blue (240), and magenta (300)

- Saturation is described as the amount of gray from 0 to 1 in the color. If the value of the saturation is smaller, the color will become white and if the value of the saturation is bigger, the color will be more saturated.

- Value can be regarded as the brightness which describes the brightness of the color from 0 to 1. If the value is bigger, the color will become white and if the value is smaller, the color will become black.

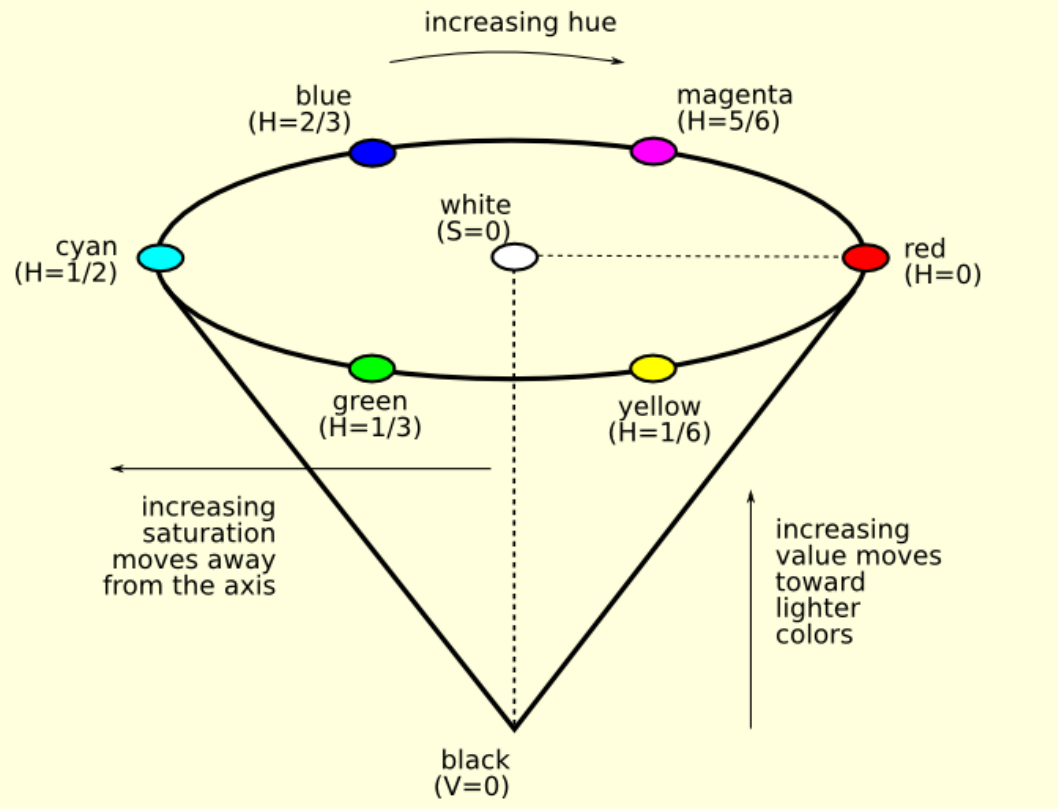

Figure 18. HSV color model /13/

- The circles which is located in the outer edge represent the H (hue) with all the colors.

- The axis of the cone is regarded as the S (saturation) which is from 0 to 1. The center of the circle means that the saturation is 0 (white) and the edge of the circle means that the saturation is 1.

- The center line of the cone is regarded as the V (Value) which is from 0 (balck) from 1 (white).

4.4 Getting Image from the NAO Robot

In order to detect the stair, it is necessary to capture the image from the camera of the NAO robot firstly. In this thesis, the bottom camera was used to get the image.

Moreover, by using Python, there are two ways to get image from the NAO robot. The one way is using ALPhotoCapture module and the other way is using ALVideoDevice module.

There are a few differences between ALPhotoCapture module and ALVideoDevice module.

4.4.1 ALPhotoCapture module

By using ALPhotoCapture module, the NAO robot can take pictures and save the pictures on the disk.

This module has two methods to take picture.

- takePicture(): This method can capture and save one picture by using the parameters which are resolution, color space, frame rate, picture format. Moreover, these parameters have set methods in order to change these values. In addition, it also needs to give the arguments of location of the file on the disk.

- takePictures(): This function is similar to takePicture(). The difference between TakePicture() and TakePictures() is that takePictures() can take many images. The names of the image files are named by appending the number of the image. /14/

Here are the default parameters of function takePictures()

Table 6. Parameter of function takePictures()

Parameter	value
Camera ID	default (top camera)
Color space	RGB
Time between two pictures	200 millisecond (5 images/second)
Resolution	VGA (640*480px)
Picture format	jpg

- Performances and limitations: The advantages of using ALPhotoCapture module are that it can take many images and it is relatively faster than ALVideoDevice module. The disadvantages of using ALPhotoCapture module is that it needs to use the absolute path of the image and cannot save the image on the remote disk.

- Example code of taking pictures. This code creates the photoCaptureProxy which is the object of ALProxy and can access to ALPhotoCapture module. Then, this code sets the resolution (640*480px) and picture format ("jpg" format) of the image. Finally three pictures of what the NAO robot sees are taken and these three picture are saved to the disk inside of the NAO robot (the path is /home/nao/recordings/cameras/). The file names are "image_0.jpg", "image_1.jpg" and "image_2.jpg".

```
# This code is used to take 3 pictures by using ALPhotoCapture module
import os
import sys
import time
```

```
from naoqi import ALProxy
IP = "192.168.1.105"    #The IP address of NAO robot
PORT = 9559
# Create a proxy to ALPhotoCapture module
try:
    photoCaptureProxy = ALProxy("ALPhotoCapture", IP, PORT)
except Exception, e:
    print "Error when creating ALPhotoCapture proxy:"
    print str(e)
    exit(1)
# Take 3 pictures in VGA and save them in the path
#(/home/nao/recordings/cameras/)
photoCaptureProxy.setResolution(2)# the resolution is 640*480px
photoCaptureProxy.setPictureFormat("jpg")# the format of picture is "jpg"
photoCaptureProxy.takePictures(3, "/home/nao/recordings/cameras/", "image")
# This call returns ['/home/nao/recordings/cameras/image_0.jpg',
#'/home/nao/recordings/cameras/image_1.jpg',
#'/home/nao/recordings/cameras/image_2.jpg']
```

Code 1. Save picture by using ALPhoteCapture module

4.4.2 ALVideoDevice Module

By using ALVideoDevice module, image can be received not only from the camera of the NAO robot but also from the simulator. Moreover, ALVideoDevice supports multi-camera. It also supports using a relative path and saving the images to a remote device. That is the main reason why the ALVideoDevice module was chosen to save images and ALVideoDevice module could be used to save images to my current workplace (the folder of my source code).

The ALVideoDevice module is in charge of the video source. Here is the process of image processing in ALVideoDevice module. /15/

- Opening the camera device and communicating through I2C bus.
- Launching the V4L2 driver in the streaming mode and a circular buffer of n elements to grab the video stream will be created by the V4L2 driver.
- Receiving subscriptions from Vision Modules by creating an ALImage for each one to encapsulate the raw data buffer and an ALImage will save the converted data.
- Selecting pictures to provide to a vision module when it requests for one, relying on whether the similar kind of picture have been requested by the other vision modules recently or not.
- Managing the whole modifications which are requested to the video device or by a V.M. on its parameters like a new color space.
- When the certain vision module unsubscribes, it will suppress corresponding unused ALImages, and check the new optimal settings for the video device and driver for the remaining modules.
- If no more vision module are subscribed, the video device will stop and close .

Here is the architecture overview of ALVideoDevice module.

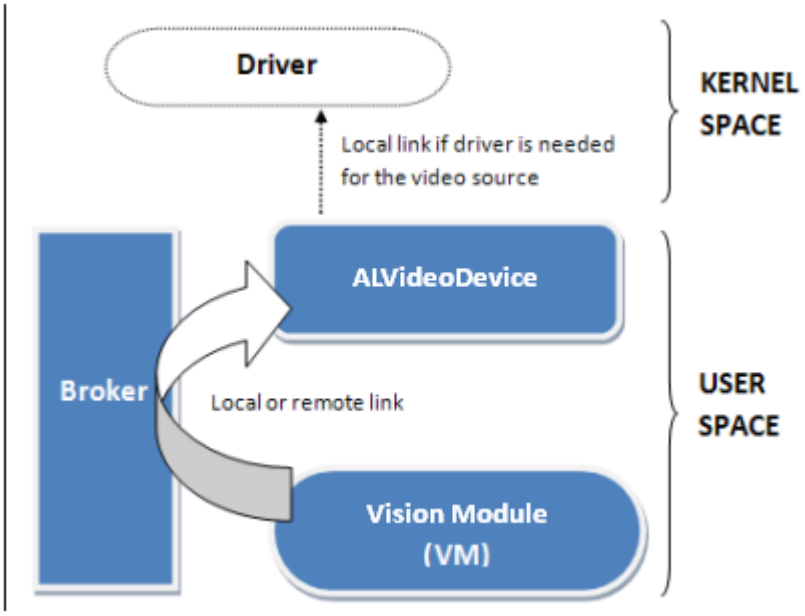

Figure 19. Architecture overview of ALVideoDevice module

A Vision Module needs to use a specific image format to perform its processing. So it subscribes to ALVideoDevice that will apply transformations on the stream to the required format which are the resolution and the color space. If the format is the native one of the video sources, a direct raw access will be requested as an advanced feature. /8/

Here is a piece of code for saving the image by using the ALVideoDevice module

```
cam = ALProxy("ALVideoDevice", IP, PORT)

resolution = 2  # VGA (640*480px)

colorSpace = 11  # RGB



videoClient = cam.subscribe("python_client", resolution, colorSpace, 5)

t0 = time.time()

# get image from camera.

naoImage = cam.getImageRemote(videoClient)
```

```
t1 = time.time()
# Time the image transfer.
print "acquisition delay ", t1 - t0
cam.unsubscribe(videoClient)
# Width is the width of the image.
Width = naoImage[0]
# Height is the height of the image.
Height = naoImage[1]
# image[6] is  an array of ASCII chars which is used for saving image data.
array = naoImage[6]
# PIL Image is created by using fromstring() method
image = Image.fromstring("RGB", (Width, Height), array)
# Save the image as a “PNG” format image
image.save(picture_name, "PNG")
```

Code 2. Save image by using ALVideoDevice module

In this code, the image is saved to the same folder of this Python code and by using the ALVideoDevice module, an absolute path of the image does not have to be provided. Thus, it will be convenient and code portability for other Python code to read and modify the data of image even on other computers or other operating systems.

4.5 Cutting Image

When the NAO robot looks down on the stair, it will see its hands and feet, the color of which is white. Thus, it will influence the NAO robot to detect white paper a lot. Meantime, NAO has almost the same standing posture every time it looks at the stairs. So the location of NAO’s hands and feet is also fixed in its sight of stairs. In addition, when the NAO robot would like to adjust the angle and distance between

one stair and itself, it just needs to focus on only one stair and the other stairs do bit need to be seen at all.

Therefore, cutting image will be the best solution to avoid this noise of picture and the influence of other stairs, its hands and feet.

Here is the image before cutting.

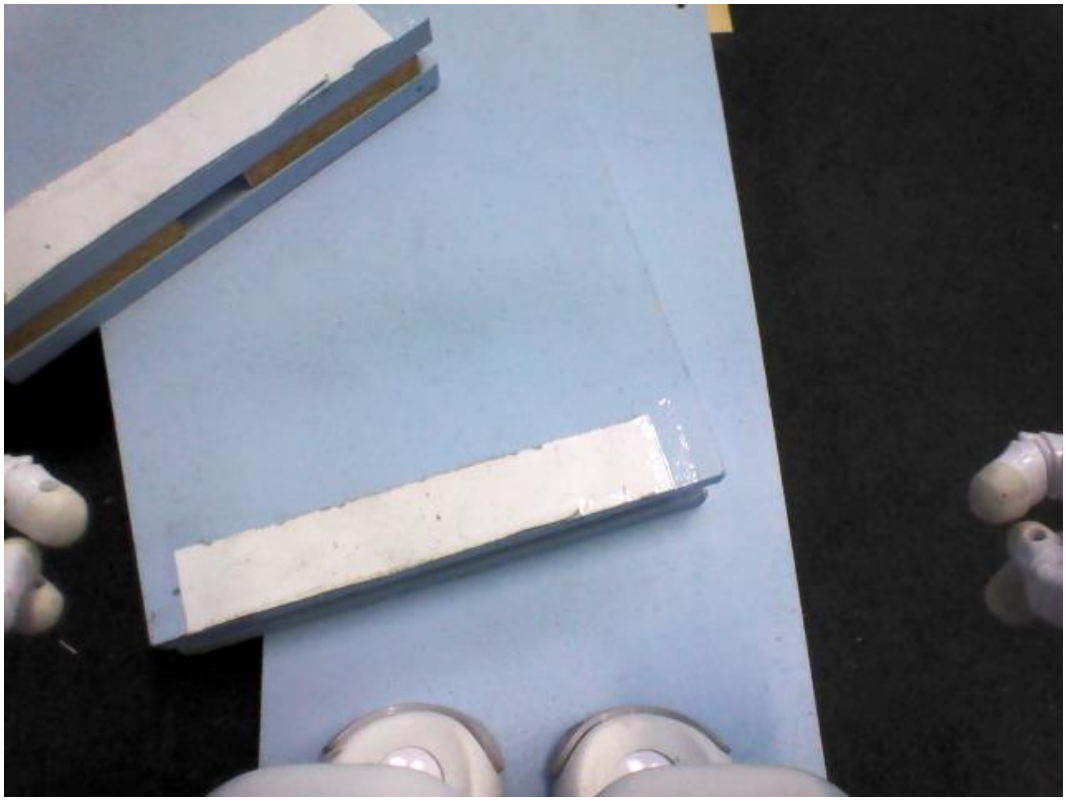

Figure 20. The image before cutting

Here is the image after cutting

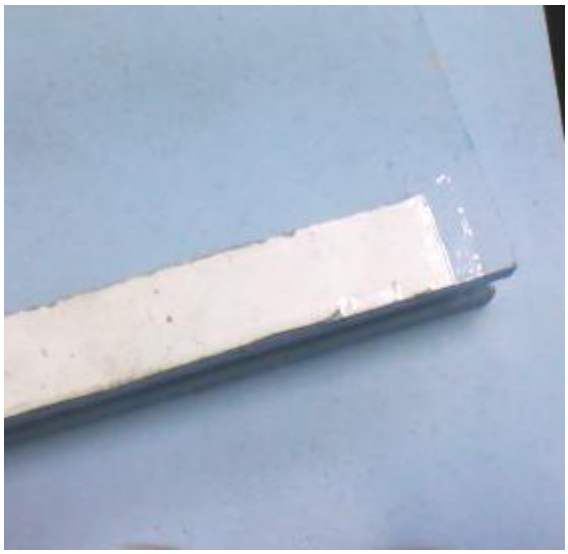

Figure 21. The image after cutting

When comparing these two pictures, the image after cutting will be more easy, effective and efficient to get the correct information of write paper.

Here is a piece of code for cutting image.

```
def cutImage(picture_name, y1, y2, x1, x2):
    #read image by using opencv
    img = cv2.imread(picture_name)
    #cut image
    img = img[y1:y2, x1:x2]
    #save the cutting image
    cv2.imwrite(picture_name, img)
```

Code 3. Cut image

4.6 Detect White Paper and Red Paper

After cutting the image, the robot needs to detect the stair in the picture. In this thesis, one white rectangle paper was used to represent one stair and the red rectangle paper was used to represent the end position. In this case, detecting the

white rectangle paper can be regarded as detecting the stair. Detecting the red rectangle paper can be regarded as detecting the end position.

Moreover, detecting white paper was used for finding the stair so that the NAO robot can get the angle and distance between the stair and itself. Detecting red paper was used for finding the end position so that the NAO robot can get the angle and distance between the end position and itself.

4.6.1 Convert RGB to HSV and Algorithm

Because detecting white paper and red paper by using the RGB (red, green and blue) color space is inaccurate. The image needs to convert its color space from RGB color space to HSV color space which is much easier to find a certain color than the RGB color space.

Here is the algorithm of converting RGB to HSV /16/

In case of 8-bit and 16-bit images, red, green and blue are converted to the floating-point format and scaled to fit the 0 to 1 range.

$$\mathrm{V} = \max(\mathrm{R}, \mathrm{G}, \mathrm{B}) \tag{1}$$

$$\mathrm{S} = \begin{cases} \frac{V-\min(R,G,B)}{V} & if(V \neq 0) \\ 0 & if(V = 0) \end{cases} \tag{2}$$

$$\mathrm{H} = \begin{cases} \frac{60(G-B)}{V-\min(R,G,B)} & if(V = R) \\ 120 + \frac{60(B-R)}{V-\min(R,G,B)} & if(V = G) \\ 240 + \frac{60(R-G)}{V-\min(\mathrm{R},\mathrm{G},\mathrm{B})} & if(V = B) \end{cases} \tag{3}$$

$$\text{If } (\mathrm{H}<0),\ \mathrm{H} = \mathrm{H} + 360 \tag{4}$$

In case of this formula, the output

- $0 \leq \mathrm{V} \leq 1$ (5)
- $0 \leq \mathrm{S} \leq 1$ (6)

- $0 \leq H \leq 360$ (7)

Then, convert these value to the value of certain data type

- 8 bit image:

  $V = 255V$ (8)

  $S = 255S$ (9)

  $H = H/2$ (10)

- 16 bit image

  $V = -65535V$ (11)

  $S = -65535S$ (12)

  $H = -H$ (13)

4.6.2 Result for Find White Paper

Figure 22. The original picture of white paper

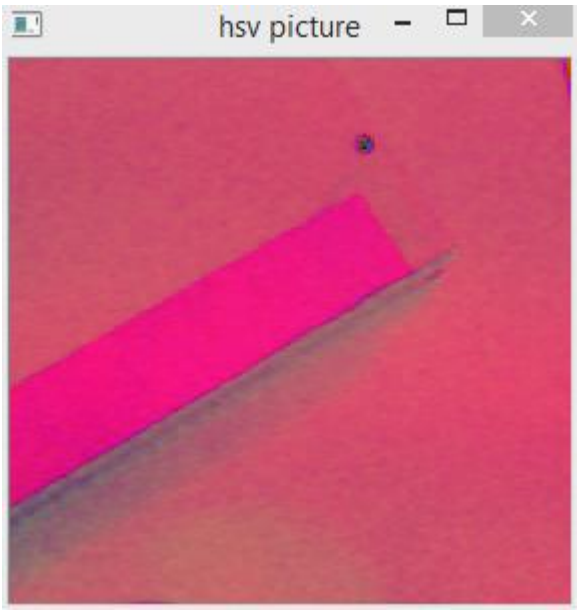

Figure 23. The hsv picture of white paper

Figure 24. The picture of finding white paper

As shown in the Figures 22, 23 and 24, detecting the white rectangle paper can roughly recognize the location and the shape of the stair.

4.6.3 Result for red paper

Figure 25. The original picture of red paper

Figure 26. The HSV picture of red paper

Figure 27. The picture of finding red paper

As shown in Figures 38, 39 and 40, detecting the red rectangle paper can roughly recognize the location and the shape of the end position.

4.7 Image Filtering

As shown in Figures 22-27, there are still some black holes inside the white rectangle paper and red rectangle paper. It is still difficult to calculate the angle and distance.

Therefore, these pictures need to be filtered in order to get more precise image. By using dilation and erosion, the black hole inside the paper can be removed.

4.7.1 Dilating Image

Dilating image is a method that only accepts a black and white picture. It turns on pixels which were near pixels that were on originally, thereby thickening the items in the image.

Dilating image can be used for connecting some discrete and disconnected pixels.

4.7.2 Eroding Image

Eroding image is the sister method to dilating image. It turns off pixels which were next to pixels that were off originally. Thus it takes away at the edges of the items in the image.

Eroding image can be used for removing the edges of pixels in the image.

Combining dilating image with eroding image (First dilating image and then eroding image) can not only connect the disconnected pixels but also keep the size and shape of the graph in the image.

4.7.3 Result after Dilation and Erosion

Here are the result after dilating image and eroding image.

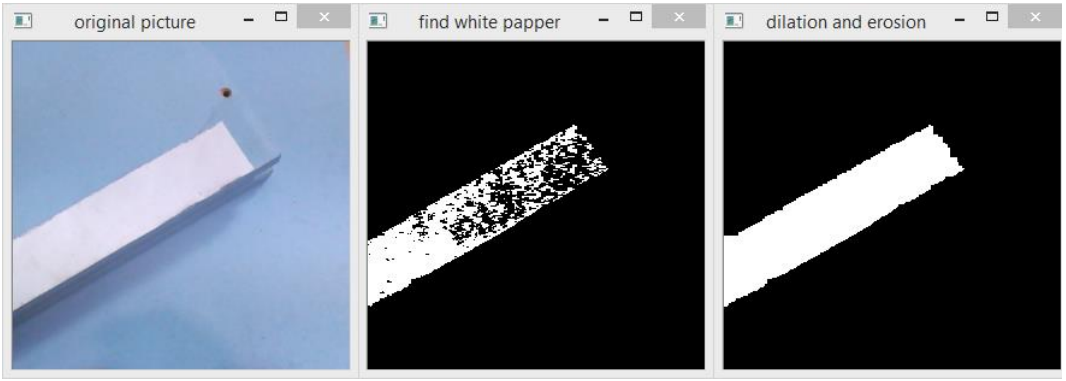

Figure 28. Dilation and erosion of image (white paper)

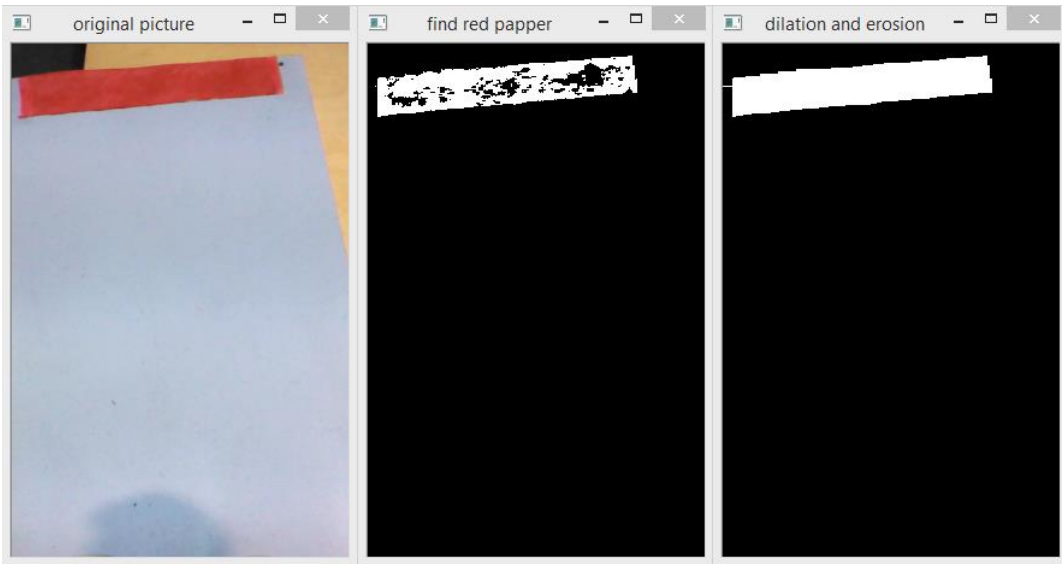

Figure 29. Dilation and erosion of image (red paper)

As shown in these pictures, the shape, position and size can be detected successfully.

4.8 Finding Contours

After white paper and red paper were found successfully, next step was to find contours and get the information of the contours which are the pixels coordinates of the contours in order to calculate the angle and distance.

4.8.1 Introduction to Finding Contours

Contours can be simply regarded as a curve which contains all the continuous points (pixels) along the boundary, having the same color or intensity.

Finding contours can be regarded as finding white objects whose background color must be black. Thus, before finding contours, the color space of the image need to convert from RGB color space to gray color space.

After finding contours, cv2.findContours() function will return the Numpy array which contains all the pixel coordinates of the contours.

4.8.2 The Result

Here is the picture of finding contours of one stair. The green line is the contours of the stair.

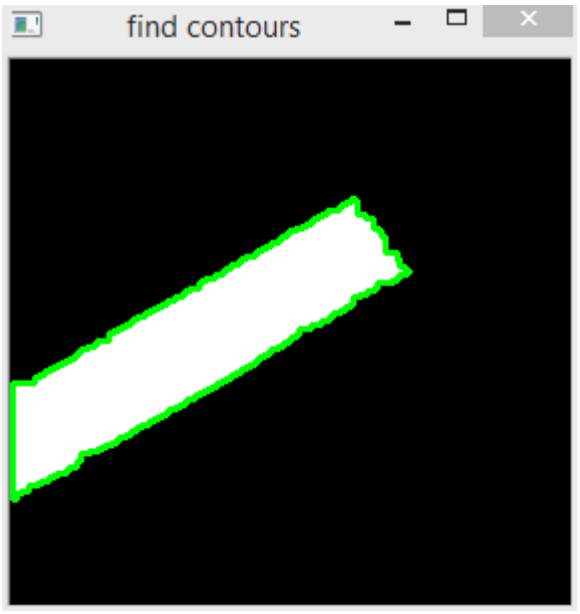

Figure 30. Finding contours of one stair

After finding contours, the program will return the pixel coordinates of all the contours. The calculation of these pixel coordinates will be described in Chapter 4.

Here are the pixel coordinates:

[array([[[170, 69]], [[170, 70]], [[169, 71]], [[168, 71]],

[[167, 72]], [[166, 72]], [[165, 73]], [[164, 73]],

[[164, 74]], [[163, 75]], [[159, 75]], [[157, 77]],

[[156, 77]], [[155, 78]], [[154, 78]], [[153, 79]],

[[152, 79]], [[149, 82]], [[148, 82]], [[147, 83]],

[[146, 83]], [[145, 84]], [[143, 84]], [[142, 85]],

```
[[140,  85]],      [[139,  86]],      [[138,  86]],      [[138,  87]],

[[137,  88]],      [[136,  88]],      [[135,  89]],      [[134,  89]],

[[134,  90]],      [[133,  91]],      [[130,  91]],      [[130,  92]],

[[129,  93]],      [[128,  93]],      [[127,  94]],      [[126,  94]],

[[125,  95]],      [[124,  95]],      [[123,  96]],      [[122,  96]],

[[122,  97]],      [[121,  98]],      [[118,  98]],      [[117,  99]],

[[116,  99]],      [[113, 102]],      [[110, 102]],      [[110, 103]],

[[109, 104]],      [[108, 104]],      [[107, 105]],      [[106, 105]],

 …                  …                  …                  …

[[184,  85]],      [[183,  84]],      [[182,  84]],      [[181,  83]],

[[181,  79]],      [[178,  79]],      [[177,  78]],      [[177,  77]],

[[174,  77]],      [[173,  76]],      [[173,  70]],      [[172,  70]],

[[171,  69]]])]
```

Figure 31. The result of finding contours

Here is the code for finding contours

```
#read image by using opencv
img=cv2.imread(picture_name)
#convert color space from RGB to GRAY color space
imgray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#draw contours
ret,thresh = cv2.threshold(imgray,127,255,0)
contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(img, contours, -1, (0,255,0), 2)
```

Code 4. Find contours

5 BEHAVIOR DESIGN

As for the behavior design, Choregraphe 2.1.2 and Webots were used to design the behavior of climbing the stairs.

5.1 Timeline Box

The timeline box contains the timeline which can synchronize boxes with movements, movements with each other and boxes with each other.

As for the timeline, the frame is the unit of a timeline. Each frame has a number of frame which corresponds to its position in the Timeline. The frame includes three type of frame which are start frame, end frame and motion key frame.

- Start frame is a specific frame which is located at the first frame of the timeline.
- End frame is a specific frame which is located at the end frame of the timeline.
- Motion key frame is represents a position and posture of the robot or a part of its body.

In the project, the rate of frame was set to 25 frames per second (FPS). Resources acquisition was set to the passive mode which means that the motion will be done if the NAO robot can be done, otherwise, the NAO robot will not done.

5.2 Key frames

In this thesis, key frames are used for designing the behavior of climbing the stairs. By using key frames, the parameters of all the joints of the NAO humanoid robot do not have to be paid too much attention to. The only thing which need to do is to make the robot do some key postures by hand and save these key postures to key frames at the certain time point. The robot will calculate how to move between two key frames automatically. Therefore, by connecting all the key frames, the behavior of climbing the stairs can be built quickly.

.

Here are the some of the key frames for designing the animation of climbing the stair.

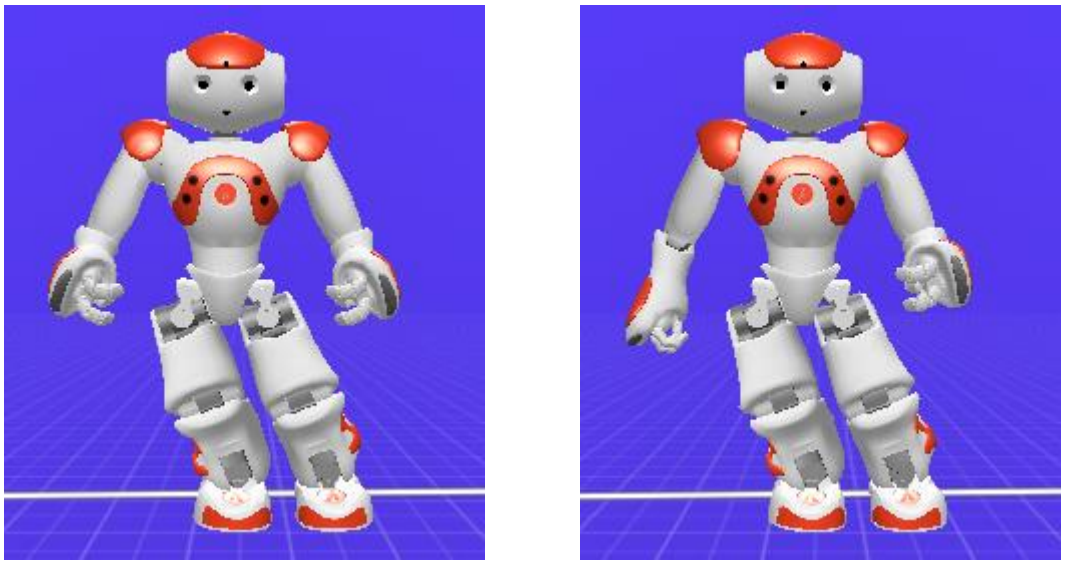

Figure 32. Key frames 50 and 60

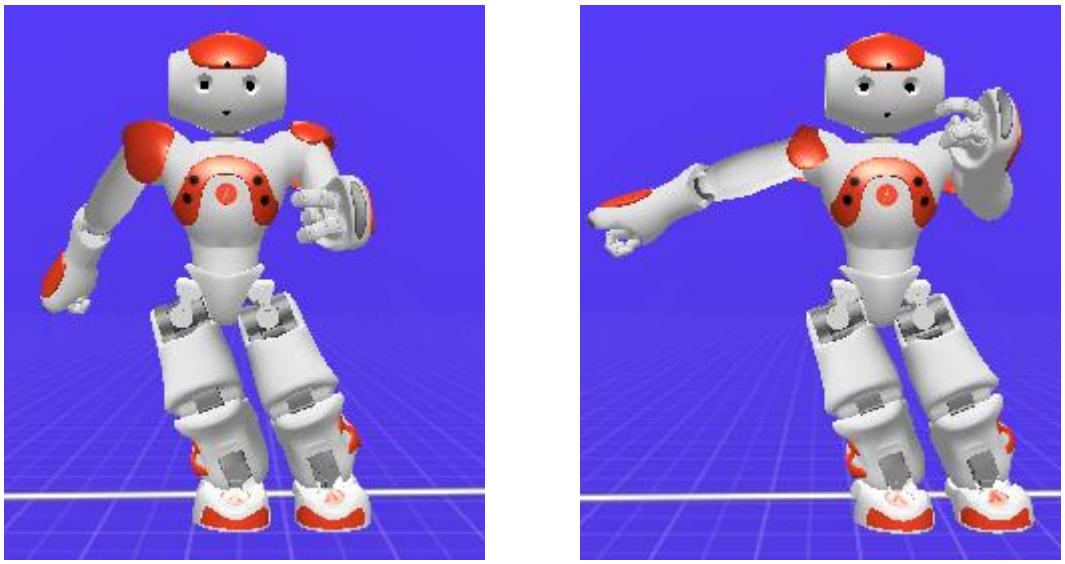

Figure 33. Key frames 72 and 90

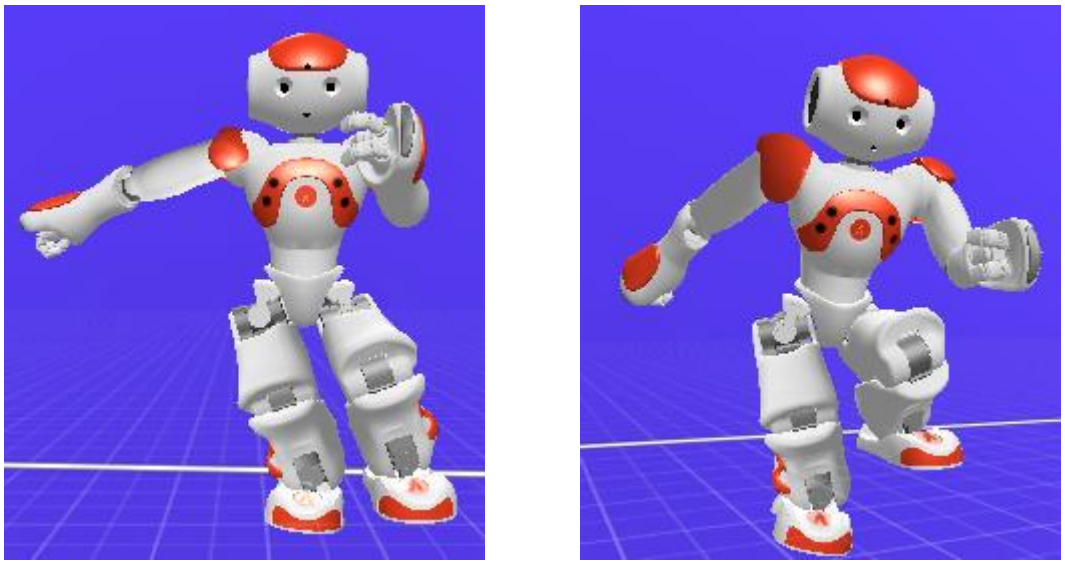

Figure 34. Key frames 116 and 210

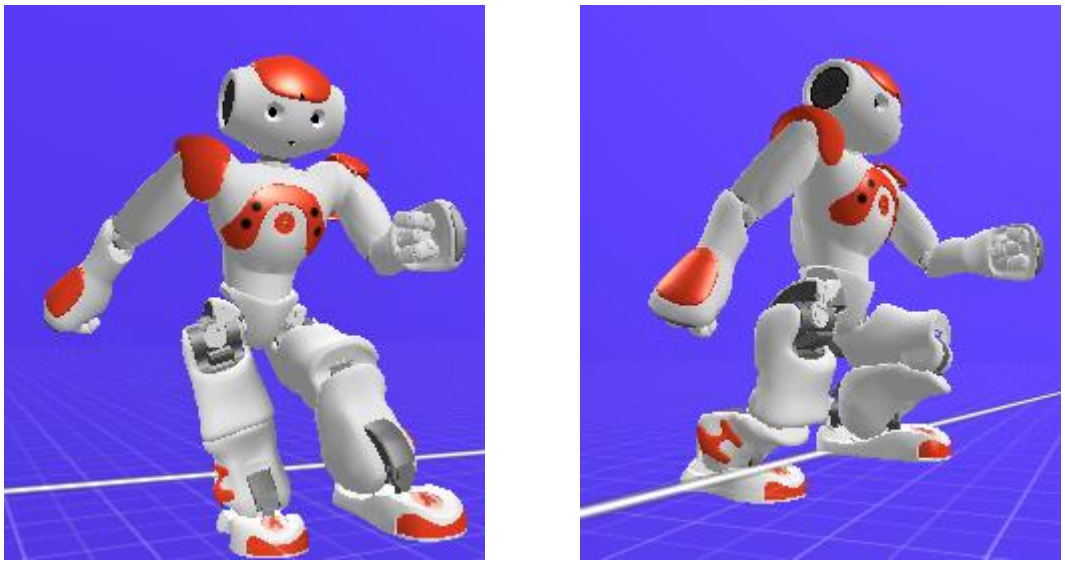

Figure 35. Key frames 370 and 510

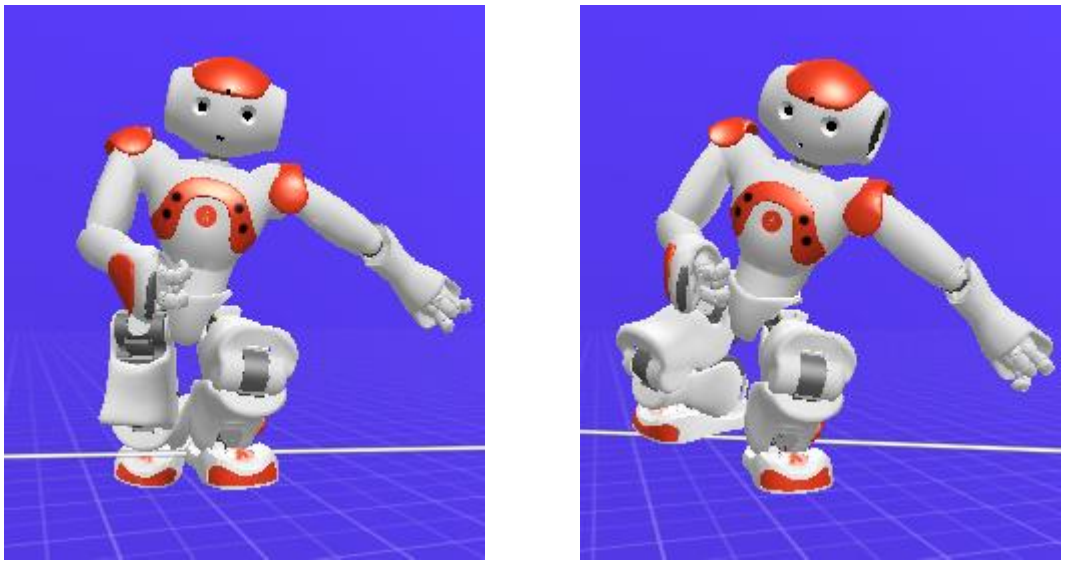

Figure 36. Key frames 550 and 670

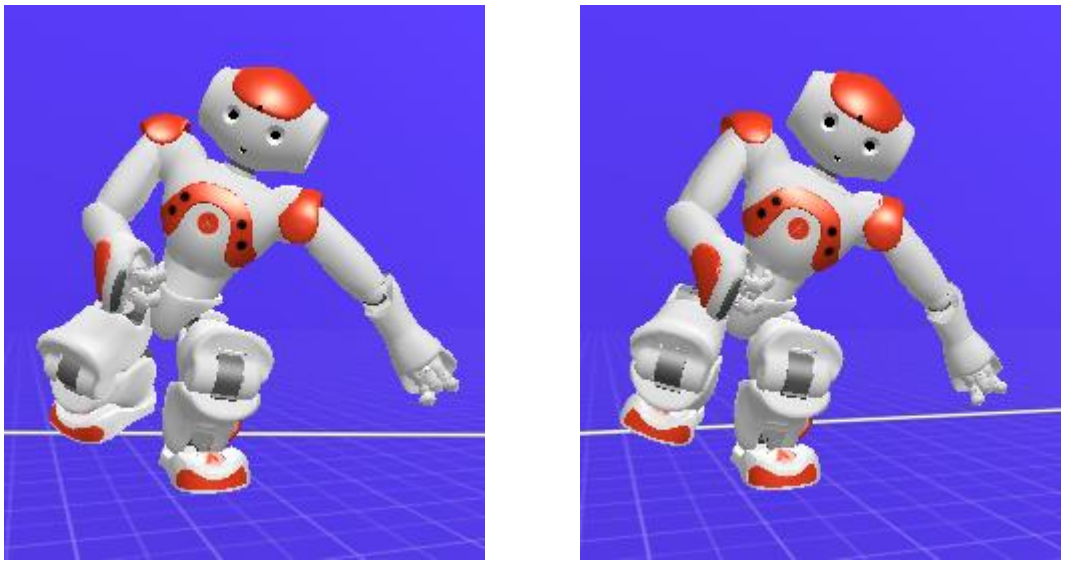

Figure 37. Key frames 700 and 750

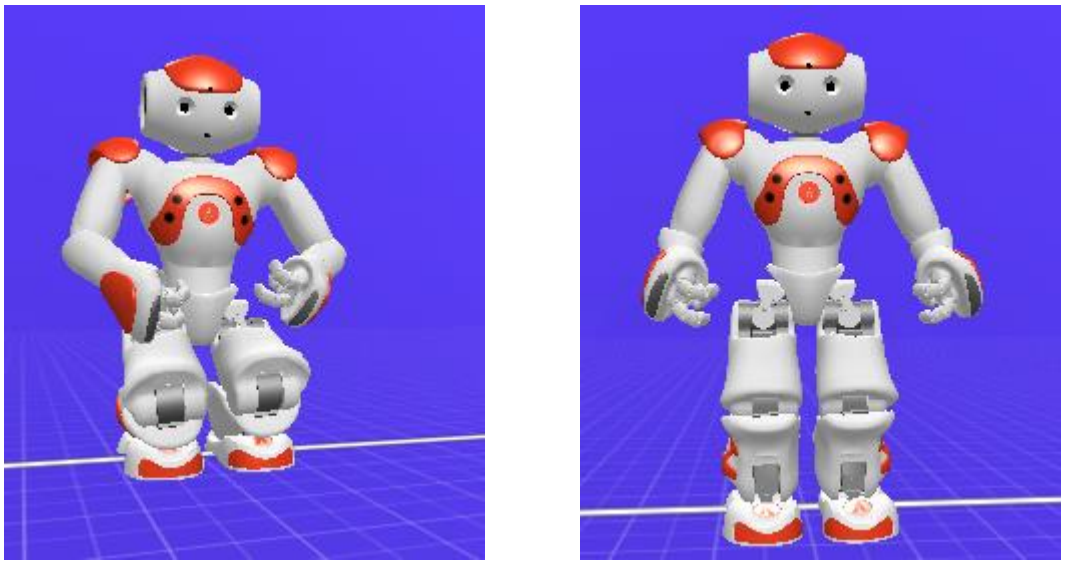

Figure 38. Key frames 810 and 850

5.3 Simulation

After designing the animation of climbing the stair, the animation cannot always be tested with a real robot because some behaviors may be harmful to the robot. Therefore, simulation for NAO robot is needed for this thesis.

Webots is a robot simulator which can be used to test the behavior of the NAO robot because testing with real robot directly may be harmful to the NAO robot. Moreover, if a real robot is used too much, the motors of the real robot will become hot soon and then the robot needs to rest around an hour. Therefore, testing behaviors of the NAO robot with Webots is more effective and efficient.

Moreover, Webots 8.0.3 also supports the gravity system so that if the virtual NAO robot could not keep its balance, the virtual NAO robot would fall down just like the real NAO robot.

6 STRATEGY

6.1 Calculation of the Steps of Staircase

When the NAO robot detects the white rectangle paper of the whole staircase, it will find all the contours (white objects) of the whole picture. In Figure 34, three stairs and some noises which are some small dots can be easily identified.

The biggest difference between the stairs and noises is that the stair has more pixel coordinates than the noises.

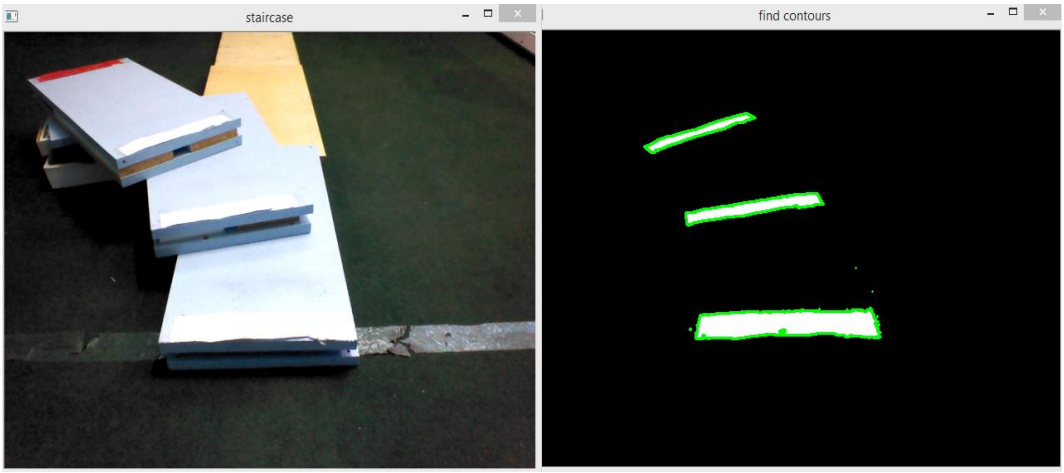

Figure 39. Detect the staircase

Here is a piece of code which is used to identify the stair and calculate the steps of the staircase.

```
#initial the steps with 0
stair=0
#read picture by using opencv
im=cv2.imread(picture_name)
#convert the color space from BGR to Gray
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray,127,255,0)
#find the contours
contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
#draw the contours by using green line
cv2.drawContours(im, contours, -1, (0,255,0), 2)

length_total = len(contours)
```

```
print "total length is",length_total
for i in range(0,length_total):
    print len(contours[i])
    if(len(contours[i])>30):
        stair=stair+1
print "----------------"
print "stair = ",stair
word = "It has "+str(stair)+" steps."
tts = ALProxy("ALTextToSpeech",IP, PORT)
tts.say(word)
return stair
```

Code 5. Calculate the steps of the staircase

In this code, the threshold was set to 30 which means that if the amount of the pixel coordinates of the white objects is more than 30, this white objects will be regarded as one stair, and otherwise, the white objects will be regarded as noises. After getting the steps of the staircase, this function will return the value of the steps

Here is the result for finding all the contours

```
total length is 15
1
1
2
1
1
2
1
172
6
4
7
1
1
117
164
----------
stair =  3
```

Figure 40. Result of calculating steps

This result shows that there are 15 white objects in the picture, but only three white objects contain more than 100 pixel coordinates and the other white objects have less than 10 pixel coordinates. It is because the contours of the stairs contain more pixel coordinates than the small noises. Therefore, setting a proper threshold can be

used to recognize the stairs and the noises. In this thesis, the threshold is set to 30. That is why this staircase has only three steps.

6.2 Checking Stair

Every time when calculating the angle and distance between the stair and the NAO robot, checking the stair is necessary for the NAO robot so that the NAO robot can get the right value of the angle and distance.

Checking the stairs is to check which contour of the white paper has more than 15 pixel coordinates, which is the same technology as calculating the steps of the staircase.

6.3 Calculation of Angle

In order to climb the stair successfully, the NAO robot need to rotate its body directly to the stair.

First of all, the NAO robot gets all the pixel coordinates of the contour of one stair by using “Find contour” method. These pixel coordinates start from the top pixel coordinate, the other pixel coordinates are found in counterclockwise direction and go back to the top pixel coordinates.

As for calculating the angle, two points, A which is the left bottom point and B which is the right bottom point are used to calculate the angle. As shown in Figure 36, the pixel coordinate of A is (x1, y1) and the pixel coordinate of B is (x2, y2).

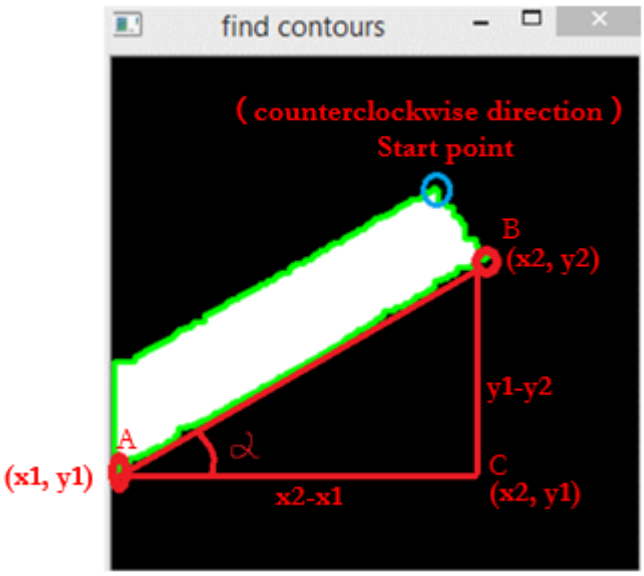

Figure 41. Calculate the angle

So,

$$BC = y1 - y2 \tag{14}$$

$$AC = x2 - x1 \tag{15}$$

$$\tan\alpha = \frac{y1-y2}{x2-x1} \tag{16}$$

$$\alpha = \arctan(\tan\alpha) = \arctan\frac{y1-y2}{x2-x1} \tag{17}$$

This angle α is that the NAO robot need to rotate.

6.4 Calculation of Distance

Here is a picture of the stair, and with a lot of experiments, it was found that when the bottom stair was at the y=230, the NAO robot could climb the stair successfully.

Figure 42. Calculating the distance

Therefore, before calculating the distance between the NAO robot and the stair (white rectangle paper), it is necessary to find the relationship between real distance and distance of pixel coordinates.

Because the NAO robot is very close to the stair and it just need to move less than 2 centimeters. Linear function is good enough to get the relationship between real distance and distance of pixel coordinates.

Here is the picture which is used modeling the relationship between real distance and distance of pixel coordinates.

Figure 43. Real distance and pixel coordinate distance

In this picture, 0cm is at the 158 of the pixel coordinates

Table 7. Modeling distance table

Pixel coordinates distance	Real distance
158-158=0	0cm
172-158=14	1cm
186-158=28	2cm
200-158=42	3cm
216-158=58	4cm
230-158=72	5cm

- For real distance between 0cm and 1cm

  The slope $k = \frac{0.01-0}{14-0} = 0.000714$ (18)

- For real distance between 1cm and 2cm

  The slope $k = \frac{0.02-0.01}{28-14} = 0.000714$ (19)

- For real distance between 2cm and 3cm

  The slope $k = \frac{0.03-0.02}{42-28} = 0.000714$ (20)

- For real distance between 3cm and 4cm

  The slope $k = \frac{0.04-0.03}{58-42} = 0.000625$ (21)

- For real distance between 4cm and 5cm

  The slope $k = \frac{0.05-0.04}{72-58} = 0.000714$ (22)

Here is picture showing the relationship between the real distance and the pixel coordinate y.

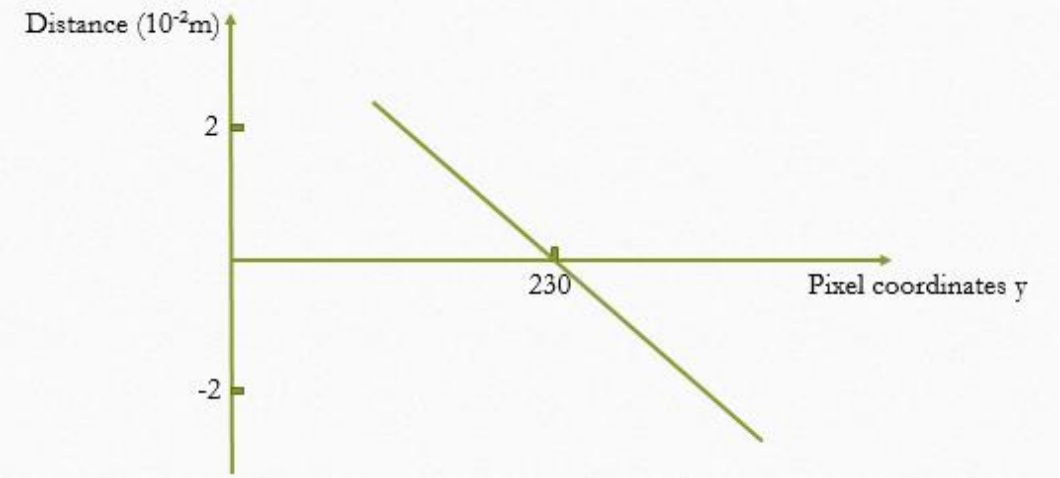

Figure 44. Real distance and pixel distance

The slope is almost the same, so the relationship between the real distance and the distance of pixel coordinates can be regarded as the linear relationship

The average of slope k

$$= \frac{(0.000714 + 0.000714 + 0.000714 + 0.000625 + 0.000714)}{5}$$

$=0.0006962 \approx 0.0007$

So,

The relationship is,

$$\text{Distance of pixel coordinates} = 230 - y \quad (23)$$

$$\text{Real distance} = (230 - y) * 0.0007 \quad (24)$$

The unit of real distance is meter (m).

7 TROUBLESHOOTING

7.1 Detecting Stair

There are two methods for detecting the stair, the one method is detecting the stair by the shape of the stair, and the other method is detecting the stair by the color the white rectangle paper.

7.1.1 Detect Stair by Shape

First, the method of detecting stair by the shape of the stair was used in the project. However, detecting the stair by shape was not accurate because the robot could not get the accurate distance between the stair and itself.

Here are the steps of detecting stair by shape.

- Get the image from the bottom camera of the NAO robot.
- Use Otsu's method to reduce the noise.
- Find the contours of the stairs.
- Check which contour is the stair.
- Calculate the angle.
- Calculate the distance

Here is the picture of detecting stair by shape.

Figure 45. Detecting the stair by shape

As shown in Figure 39, although it can roughly show the stairs, this picture still has so many big noises and it is difficult to find the stairs.

Moreover, if the shadow of NAO robot is on the stair, the shadow will influence detecting the stair and the NAO robot will think that the shadow is not the stair.

In this case, the stair which the NAO robot detects is not the integrated stair. The robot cannot get the correct angle and distance between the NAO robot and the stairs.

Here is the original picture of the stair where the shadow is on the stair.

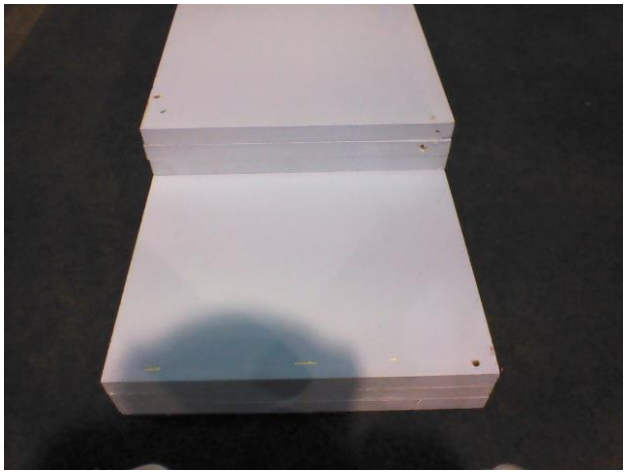

Figure 46. The stair with the shadow.

Here is the processed picture of the stair with the shadow.

Figure 47. Processed picture of stair with shadow

As shown in Figure 40 and Figure 41, it is difficult to calculate the angle and distance by using this picture.

In addition, if the light-intensity is not strong enough, detecting stair by shape cannot work anymore.

Therefore, detecting the stair by the shape is not accurate. That's why detecting the stair by color was used in this thesis.

7.1.2 Detect Stair by Color

Since detecting stair by the shape of the stair is not accurate, detecting the color of white rectangle paper which is regarded as a stair was used instead.

First, the color space of the image is RGB (red, green and blue). Although RGB can describe any color by using an ordered list which contains red, green and blue. However, if the light conditions changes, the RGB value will change a lot.

In comparison, HSV is much better at handling lighting differences and HSV is much easier to separate or compare color. Therefore, the color space of the image is changed into HSV color space. By using HSV color space, detecting stair becomes much more accurate than detecting stair by using method of detecting the shape of the stairs. /10/

Here is one example of detecting stair by color in Figure 42.

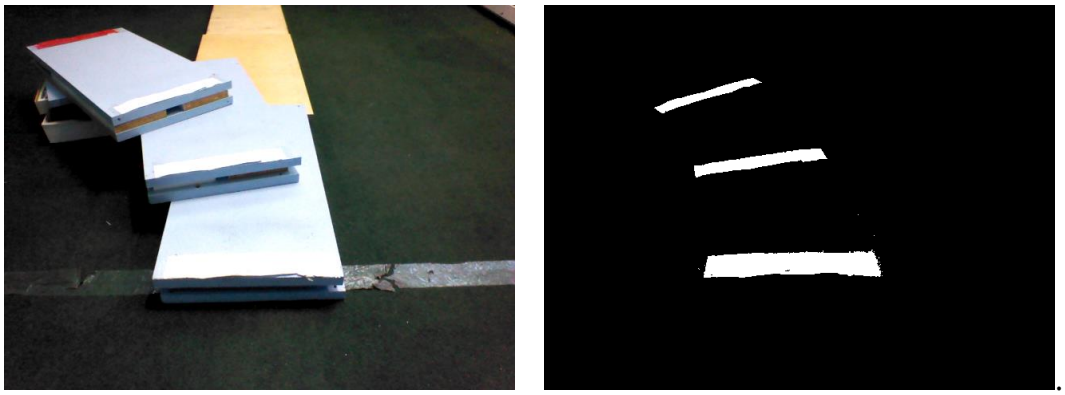

Figure 48. Detecting stairs by color

As shown in Figure 61, detecting the color of white paper is much better than detecting the shape of the stair.

7.2 Improving the Movement Method

Because the stair was constructed of some blue rectangle board and some books, the staircase was not stable enough to make the robot move well. Meanwhile, if the NAO robot loses its balance when it moves to the stair or rotates its body, the robot will stop moving and do next steps. If this case happened, the robot would not climb the stairs successfully. Therefore, it was necessary to improve the movement method.

Every time the NAO robot moves, it can record how long it moves and how many degrees it rotates. Then, these values are compared with the value which user sets. If the NAO robot does not reach the destination, it will move again till the NAO robot moves to the destination.

By using this method, the NAO robot can reach the destination.

```
realEndPosition = almath.Pose2D(motion.getRobotPosition(False))
positionError = realEndPosition.diff(expectedEndPosition)
positionError.theta = almath.modulo2PI(positionError.theta)
```

Code 6. Get the error value of movement

In Code 6, positionError is an array which contains “positionError.x”, “positionError.y” and “positionError.theta”. “positionError.x” represents the error distance in direction of forward direction. “positionError.y” represents the error distance in direction of in left-right direction. “positionError.theta” represents the error degree that the NAO robot need to rotate.

7.3 Adjust the Robot’s Head before Taking Image

When the body of the NAO robot is directly to the stair, the head of NAO robot is not directly to the stair. Therefore, sometimes, the NAO robot cannot get the accurate angle and distance.

As for angle, the head of NAO robot needs to turn left and the angle is 2 degrees. In this way, the head of NAO robot will directly face to the stair.

Here is the code for turning the head.

```
def turnHead(degree, IP, PORT):

    # Create a proxy to ALMotion.
    try:
        motionProxy = ALProxy("ALMotion", IP, PORT)
    except Exception,e:
        print "Could not create proxy to ALMotion"
        print "Error was: ",e

    # Create a proxy to ALRobotPosture.
    try:
        postureProxy = ALProxy("ALRobotPosture", IP, PORT)
    except Exception,e:
        print "Could not create proxy to ALRobotPosture"
        print "Error was: ",e
    time = 0.5
    if((degree >= -118) & (degree <=118)):
        motionProxy.angleInterpolation(
        ["HeadYaw"],
        [math.radians(degree)],
        [time],
        True
        )
```

Code 7. Turning head of NAO robot

As for the distance of pixel coordinate, it is better to use the average value of the pixel coordinate instead of one value. The left bottom point A (x1, y1) and the right bottom point B (x2, y2) of the stair were used to calculate the average value of the pixel coordinates y and pixel coordinate y equals to (y1 + y2) / 2.

8 FUTURE RESEARCH

As the NAO robot is a huge system, there are so many functions to improve and complete. The object of doing the NAO robot project was that our robotics team can participate in the RoboCup competition.

8.1 Going Downstairs Autonomously

This thesis is focuses on climbing the stairs autonomously with the NAO humanoid robot. Detecting the stair can use the same technology of climbing the stairs and use a white rectangle color paper to represent the stair. Then, the NAO robot also needs to rotate its body or adjust the distance between the stair and its body. Finally, the NAO robot can go downstairs.

8.2 Improving the Posture of Climbing the Stairs

In this thesis, although the NAO robot can climb the stairs successfully, the posture of climbing the stairs is not perfect like with the real human beings. Some postures of the NAO robot need to improve.

An example is shown in Figure 43.

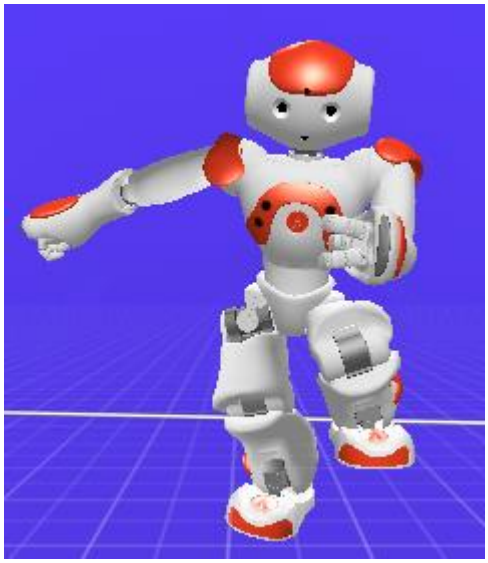

Figure 49. Not suitable posture of NAO robot

In order to solve this problem, the center of gravity must be calculated because the NAO robot not only need to do some good postures but also needs to keep the balance of its body.

If the NAO robot can keep the balance with its feet, the arm can move more smoothly like the real human beings

Moreover, the NAO robot can move its head when it is climbing the stair

8.3 Design the Behavior for Climbing the Stairs Autonomously.

In this thesis, the biggest limitation is that the height of each stair is fixed (4.7 cm). Meanwhile, the design of the behavior for climbing the stairs is based on the key frames, which means that the movement of climbing one stair is fixed. The robot just need to find the proper position so that the NAO robot can climb the stair successfully. However, if the height of the stair changed, the NAO robot would not climb the stair any more.

Thus, if the height of the stair changes, there are some suggestions for the future research.

- Calculate the height of the stair firstly. According to the height of the stair, the NAO robot can adjust the strategy for climbing the stair.
- Find a good position for climbing the stair. In this way, it will increase the probability of climbing the stair with NAO humanoid robot.
- Get real-time image of the stair so that the NAO robot can calculate its movement and know how to climb the stair.

9 CONCLUSION

This thesis introduces the design and implementation of autonomous stair climbing with the NAO humanoid robot based on its vision system, behavior system and strategy.

As for the vision system, the NAO robot has two good camera which can capture images up to 1280*960 resolution and take 30 images for one second. By using its camera, the NAO robot can calculate the angle, distance. Moreover, the NAO robot supports the external modules to make application. For example, the NAO robot can call the OpenCV module to process the image. As for the behavior design, NAO robot has 25 degrees of freedom so that it can move smoothly and do a variety of movements such as climbing the stairs, dancing and other movements. Meanwhile, the NAO robot supports the export of the animation to Python or C++ code, which helps developers to program with the animation.

All the successes depend on the hard work. In order to conduct a large project, the capability of self-motivated learning is necessary for everyone. Even though there are plenty of problems and difficulties, in fact, the number of the solutions is always bigger than the number of problems. Moreover, with finding more solutions to the problem, some better solutions will appear. For example, the method of detecting stair by color replaced the method of detecting the stair by shape. Therefore, problems and difficulties are always temporary because plenty of solutions are close to them and waiting for you.

REFERENCE

/1/ PROF. KISUNG SEO. 2013. Using NAO: Introduction to Interactive Humanoid Robots. Aldebaran Robotics.

/2/ Introduction to NAO robot. Accessed 15.04.2015

https://www.aldebaran.com/en/humanoid-robot/nao-robot

/3/ UK robots prepare for world cup. BBC News. 25 October 2010.

/4/ Aldebaran Nao Humanoid Robot. Active8 Robots UK. 2014. Retrieved 1 February2015. Accessed 15.04.2015

http://www.active8robots.com/products/nao-humanoid-robot/

/5/ Working principle of NAO robot. Accessed 18.04.2015

https://www.aldebaran.com/en/humanoid-robot/nao-robot-working

/6/ The comparison between python and C++. Accessed 20.04.2015

http://www.quora.com/What-are-the-advantages-of-Python-over-C++

/7/ two camera of NAO robot. Accessed 25.04.2015

http://doc.aldebaran.com/2-1/family/robots/video_robot.html#robot-video

/8/ Details on ALVideoDevice module. Accessed 28.04.2015

http://doc.aldebaran.com/2-1/naoqi/vision/alvideodevice-indepth.html#api-advanced

/9/ Communication of NAO robot. Accessed 29.04.2015

http://doc.aldebaran.com/1-14/dev/naoqi/index.html

/10/ Conversion of color space. Accessed 30.04.2015

http://www.shervinemami.info/colorConversion.html

/11/ The RGB color model. Accessed 01.05.2015

http://bpastudio.csudh.edu/fac/lpress/netapps/hout/rgb.htm

/12/ What is HSV color space. Accessed 01.05.2015

http://desktoppub.about.com/od/glossary/g/HSV.htm

/13/ The model of HSV. Accessed 01.05.2015

http://infohost.nmt.edu/tcc/help/pubs/colortheory/web/hsv.html

/14/ Capture pictures by using ALPhotoCapture module. Accessed 02.05.2015

http://doc.aldebaran.com/2-1/naoqi/vision/alphotocapture.html#alphotocapture

/15/ Capturepictures by using ALVideoDevice module. Accessed 02.05.2015

http://doc.aldebaran.com/2-1/naoqi/vision/alvideodevice.html#alvideodevice

/16/ Convert color space from RGB color space to HSV color space. Accessed 03.05.2015

http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html

APPENDIX 1

MAIN FUNCTION OF CLIMBING THE STAIR

```
if __name__ == '__main__':
    IP = "192.168.1.105"  # the IP address of NAO robot
    PORT = 9559         # the port of the NAO robot

    #Read IP address from first argument if any.
    if len(sys.argv) > 1:
        IP = sys.argv[1]
    motionProxy = ALProxy("ALMotion", IP, PORT)
    postureProxy = ALProxy("ALRobotPosture", IP, PORT)
    cameraModule = ALProxy("ALVideoDevice",IP,PORT)
    cameraModule.setParam(18, 1)

    #find how many steps that the robot need to climb
    head_movement.turnHead(-2.0, IP, PORT)
    head_movement.moveHead(-11.4, IP, PORT)
    imageTStair="total_stair.png"
    imageController.showNaoImage(imageTStair, IP, PORT)
    imageController.findWhitePaper(imageTStair)
    imageController.imageFiltering(imageTStair, 8)
    steps = imageController.contours(imageTStair, IP, PORT)
    print "steps=",steps

    #move to stairs
```

```
goInit.go_Init(IP, PORT)
movement.moveTo(0.32, 0, 0, IP, PORT)

#adjust degree
head_movement.turnHead(-2.0, IP, PORT)
head_movement.moveHead(22, IP, PORT)
image_stair0="stair0.png"
imageController.showNaoImage(image_stair0, IP, PORT)
imageController.cutImage(image_stair0, 154, 426, 180, 460)
imageController.findWhitePaper(image_stair0)
imageController.imageFiltering(image_stair0,5)
degree = adjust.adjustBody(image_stair0)
if(abs(degree)>0.03):
   motionProxy.moveTo(0, 0, degree-0.02)

#adjust distance
head_movement.turnHead(-2.0, IP, PORT)
head_movement.moveHead(22, IP, PORT)
image_distance="sd0.png"
imageController.showNaoImage(image_distance, IP, PORT)
imageController.cutImage(image_distance, 154, 426, 180, 460)
imageController.findWhitePaper(image_distance)
imageController.imageFiltering(image_distance,5)
distance = adjust.calculateDistance(image_distance)
goInit.go_Init(IP, PORT)
if(abs(distance)>=0.005):
```

```
    movement.moveTo(distance, 0, 0, IP, PORT)


  #climb the stair
  for i in xrange(steps):
    print "########################## STEP ",i+1, "Start ##############################"
    postureProxy.goToPosture("StandInit", 0.8)
    goUpStairs.upStairs(IP, PORT)
    if(i<steps-1):

      #adjust degree
      head_movement.turnHead(-2.0, IP, PORT)
      head_movement.moveHead(22, IP, PORT)
      image_stair="stair"+str(i+1)+".png"
      imageController.showNaoImage(image_stair, IP, PORT)
      imageController.cutImage(image_stair, 154, 426, 180, 460)
      print "steps=",steps
      if(i==0):
        imageController.findWhitePaper(image_stair)
      else:
        imageController.findWhitePaper2(image_stair)
      imageController.imageFiltering(image_stair,5)
      degree = adjust.adjustBody(image_stair)

      time.sleep(1)
      if(abs(degree)>0.03):
```

```
            goInit.go_Init(IP, PORT)


            movement.moveDegree(0, 0, degree, IP, PORT)



        #adjust distance
        head_movement.turnHead(-2.0, IP, PORT)
        head_movement.moveHead(22, IP, PORT)
        image_sd="stair_d_"+str(i+1)+".png"
        imageController.showNaoImage(image_sd, IP, PORT)
        imageController.cutImage(image_sd, 154, 426, 180, 460)
        if(i==0):
            imageController.findWhitePaper(image_sd)
        else:
            imageController.findWhitePaper2(image_sd)
        imageController.imageFiltering(image_sd,5)
        distance = adjust.calculateDistance(image_sd)
        goInit.go_Init(IP, PORT)
        time.sleep(1)


        if(abs(distance)>=0.005):
            movement.moveTo(distance, 0, 0, IP, PORT)
    print "########################## STEP ",i+1, "End ##############################"
```

```
print "############################################################"
#find red mark
head_movement.turnHead(-2.0, IP, PORT)
head_movement.moveHead(22, IP, PORT)
image_end="end.png"
imageController.showNaoImage(image_end, IP, PORT)
imageController.cutImage(image_end, 0, 426, 180, 460)
imageController.findPinkPaper(image_end)
imageController.imageFiltering(image_end, 5)
degree = adjust.adjustBody(image_end)

if(abs(degree)>0.03):
    goInit.go_Init(IP, PORT)
    movement.moveDegree(0, 0, degree, IP, PORT)

time.sleep(1)



head_movement.turnHead(-2.0, IP, PORT)
head_movement.moveHead(22, IP, PORT)
image_end="end_d.png"
imageController.showNaoImage(image_end, IP, PORT)
imageController.cutImage(image_end, 0, 426, 180, 460)
imageController.findPinkPaper(image_end)
imageController.imageFiltering(image_end, 5)
distance = adjust.calculateDistanceEnd(image_end)
```

```
goInit.go_Init(IP, PORT)
movement.moveToEnd(distance, 0, 0, IP, PORT)

tts = ALProxy("ALTextToSpeech",IP, PORT)
tts.say("I succeed!")
```